WILSON VICENTE RUGGIERO

PROJETO DE UM PROCESSADOR CENTRAL MICROPROGRAMADO

Dissertação de Mestrado apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de MESTRE EM ENGENHARIA.

Orientador: Prof. Dr. Antonio Hélio Guerra Vieira

SÃO PAULO
-1975-

DEDALUS - Acervo - EPEL



31500016338

PEL

Universidade de São Paulo
Biblioteca da Escola Politécnica

FD-87
e.2

A meus pais.

A minha esposa, de quem tanto apoio eu recebo.

A minha filha e meu irmão.

AGRADECIMENTOS

Ao prof. Dr. Antonio Hélio Guerra Vieira, pela constante orientação e incentivo.

Ao prof. Dr. Antonio Marcos de Aguirra Massola pelo apoio recebido.

Aos engenheiros Lucas Antonio Moscato, Edson Fregni e aos professores Glen George Langdon Jr., James Gregory Rudolph pelas valiosas discussões.

Aos colegas Stephan Kovach, Sérgio Gonçalves, Roger Carlos Cintra Ferreira e Edit Grassiani pelas constantes reuniões onde foram discutidos inúmeros aspectos deste trabalho.

A minha esposa Tania, pela sua dedicação e incentivo.

A srta. Marylúcia Rosa da Silva e ao Sr. Miguel Hernandez Morales pela dedicação e eficiência.

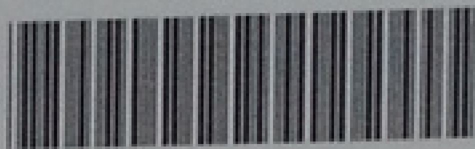
WILSON VICENTE RUGGIERO

PROJETO DE UM PROCESSADOR CENTRAL MICROPROGRAMADO

Dissertação de Mestrado apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de MESTRE EM ENGENHARIA.

Orientador: Prof. Dr. Antonio Hélio Guerra Vieira

SÃO PAULO
-1975-

DEDALUS - Acervo - EPEL

31500016338

PEL
Universidade de São Paulo
Biblioteca da Escola Politécnica

FD-87
e.2

A meus pais.

A minha esposa, de quem tanto apoio eu recebo.

A minha filha e meu irmão.

AGRADECIMENTOS

Ao prof. Dr. Antonio Hélio Guerra Vieira, pela constante orientação e incentivo.

Ao prof. Dr. Antonio Marcos de Aguirra Massola pelo apoio recebido.

Aos engenheiros Lucas Antonio Moscato, Edson Fregni e aos professores Glen George Langdon Jr., James Gregory Rudolph pelas valiosas discussões.

Aos colegas Stephan Kovach, Sérgio Gonçalves, Roger Carlos Cintra Ferreira e Edit Grassiani pelas constantes reuniões onde foram discutidos inúmeros aspectos deste trabalho.

A minha esposa Tania, pela sua dedicação e incentivo.

A srta. Marylúcia Rosa da Silva e ao Sr. Miguel Hernandez Morales pela dedicação e eficiência.

RESUMO

Para poder projetar o processador central de um minicomputador cuja arquitetura encontra-se definida são discutidos os principais aspectos da modularização de sistemas digitais e algumas sistemáticas de projeto lógico que facilitam a obtenção de um sistema flexível (facilmente adaptável a novas situações) cujo projeto seja de fácil compreensão e que utilize eficientemente os circuitos integrados MSI.

Como resultado de uma análise, cujo objetivo é a comparação das sistemáticas de projeto lógico, optou-se pelo projeto de um processador central microprogramado, sendo então descritas todas as suas principais características.

ABSTRACT

This work covers the design of the central processor of a minicomputer, the architeture of which has been previously estabilished.

Some main aspects of digital systems modularization are discussed, along with digital design practices leading to a versatile, easily understood system which makes efficient use of currently available MSI integrated circuits.

As a result of a comparative analysis of digital design practices, a microprogrammed central processor was selected, and its main features are discussed.

PROJETO LÓGICO DE UM PROCESSADOR CENTRAL MICROPROGRAMADO

I - Introdução

II - Análise das alternativas para o projeto lógico

1. Modularidade no projeto de computadores. Níveis de modularização

2. Projeto lógico com modularização em nível de transferência entre registradores. Os módulos.

3. Sistemáticas de projeto lógico

4. Projeto lógico do Fluxo de Dados

5. Projeto lógico do Controle

5.1. Projeto lógico do controle tentando a utilização de um módulo de controle universal (controle distribuído)
- O módulo de controle
- Propriedades do módulo de controle
- O projeto

5.2. Projeto lógico do controle utilizando um núcleo de controle concentrado.
- O núcleo de controle
- Propriedades do núcleo de controle
- O projeto

5.3. Projeto lógico do controle utilizando um núcleo de controle microprogramado
- Projeto das microinstruções
- Implementação das microinstruções
- O núcleo de controle microprogramado
- O projeto

6. Relação entre a complexidade do sistema e a sistemática de projeto lógico a utilizar.

III - O sistema modularizado em nível PMS

1. O sistema

2. A Via Comum

2.1. Definições
2.2. Controlador da via comum
2.3. Sinais da via comum

IV - O processador

1. Fluxo de Dados
2. Controle
2.1. Filosofia e definição da linguagem de controle
2.2. Implementação do núcleo de controle
2.3. Diagrama de tempo das microinstruções
2.4. Estados de processamento e do controle
2.5. Conjunto de microinstruções

V - Apêndice: Descrição resumida da arquitetura do processador.

1. Filosofia de proteção de memória
2. Divisão lógica do espaço de endereçamento
3. Modos de Endereçamento
3.1. Procedimento de cálculo de endereço efetivo para um deslocamento na segunda palavra das instruções longas

3.2. Procedimento de cálculo de endereço efetivo para um deslocamento contido num dos registradores de propósito geral.

4. Conjunto de instruções

I. Introdução

O tema central deste trabalho é o projeto lógico do processador central de um minicomputador. O trabalho foi motivado devido às dificuldades encontradas, no início do projeto, em adequá-lo à tecnologia de circuitos integrados MSI (Medium Scale Integration). Na pesquisa das várias soluções para o problema, encontrou-se motivação e tema suficientes para produzir esta dissertação.

Considerar-se-á conhecida a arquitetura do processador a ser projetado, a qual será descrita resumidamente no Apêndice deste trabalho. Poder-se-ia tratar, nesta dissertação, das justificativas para a arquitetura definida, mas isto estenderia por demais este relato. Por isso iremos nos deter somente no projeto lógico do processador central.

No capítulo II serão analisadas as várias alternativas para o projeto lógico de um sistema digital, tendo-se como objetivos: a realização de um sistema modular; o eficiente uso dos circuitos integrados MSI disponíveis; a facilidade de projeto e de compreensão do mesmo, e uma perfeita adequação da sistemática de projeto à complexidade do sistema a ser projetado. Ainda neste capítulo será feito uma análise comparativa entre as sistemáticas de projeto lógico apresentadas, para identificar as faixas de aplicação de cada uma delas, ou seja, dado um sistema digital de uma determinada complexidade, serão mostradas algumas das técnicas que se aplicam melhor ao projeto lógico deste sistema.

Tendo sido descritas algumas sistemáticas de projeto lógico e elaborado um critério de seleção para a aplicação das mesmas, será apresentada, no capítulo IV, a solução adotada, descrevendo-se tanto o fluxo de dados como a unidade de controle do processador. Também será apresentada no capítulo III a solução escolhida para se implementar um sistema modular em nível PMS (Processador Memória Seleção), sendo descrita a estrutura de blocos do sistema organizado em torno de uma via comum de comunicação.

Convém ressaltar que todas as soluções apresentadas nos capítulos III e IV foram implementadas eletricamente, existindo um protótipo do processador no Laboratório de Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

II. ANÁLISE DAS ALTERNATIVAS PARA O PROJETO LÓGICO

1. Modularidade no projeto de computadores

Níveis de modularização

Serão descritas, neste capítulo as alternativas que melhor se aplicam ao projeto lógico de um sistema digital, levando-se em conta a utilização eficiente de circuitos integrados MSI, a obtenção de um sistema modular e a facilidade de projeto e compreensão do mesmo.

Não serão descritas todas as alternativas possíveis para o projeto lógico, mas serão apresentadas sistemáticas que possuem um ou mais dos objetivos supra citados, dentre todas as pesquisadas na literatura especializada.

Antes de iniciar a apresentação das alternativas para o projeto lógico, convém esclarecer melhor os três objetivos procurados.

Quando se diz que se pretende a utilização eficiente de circuitos integrados MSI disponíveis, visa-se o aproveitamento do estado atual da tecnologia de circuitos integrados no projeto lógico.

Desta forma consegue-se compactar o sistema, obtendo para uma mesma capacidade um tamanho menor, comparado a uma implementação usando somente circuitos integrados SSI (Single Scale Integration).

Para se ter uma sistemática de aplicação simples, deve-se utilizar um procedimento de transformação direta (relação um para um) entre uma maneira fácil e consagrada de se descrever um algorítmo (diagrama em blocos) e o circuito realizador da função desejada.

As sistemáticas de projeto lógico, que permitem obter, de um determinado fluxograma que descreve o comportamento do circuito, o próprio circuito através de uma manipulação simples, acabam gerando também um circuito de fácil entendimento.

Para poder analisar o projeto de computadores e sistemas digitais tentando evidenciar aspectos que tornem conveniente a sua modularização, deve-se definir precisamente o que se entende por um sistema modular.

Um sistema modular (1) é constituído por um agrupamento de unidades padrão, chamadas módulos, interligados segundo um conjunto de regras, que padronizam a comunicação entre módulos, de maneira a obter flexibilidade na realização de um determinado algorítmo.

Pela definição acima apresentada, pode-se perceber a atração que a modularização exerce sobre os projetistas de computadores e sistemas digitais. A grande flexibilidade apresentada por sistemas modulares, certamente é a característica que torna este objetivo um dos principais no projeto do processador.

Um sistema de computação pode possuir diferentes aspectos na sua modularização. Pode-se ter tanto o "software" como o "hardware" modularizados. Apesar da grande importância da modularização do "software" de um sistema, foge aos objetivos deste trabalho a sua análise. Consideraremos somente os diversos níveis de modularização no "hardware" do sistema.

O "hardware" de um sistema digital pode apresentar modularização em diferentes níveis, dependendo das unidades funcionais escolhidas para formar o conjunto de módulos padrão. No caso de um computador digital pode-se identificar facilmente dois níveis de modularização.

O primeiro deles possui um conjunto de módulos padrão com blocos funcionais tais como processador central, canal de entrada e saída de dados, memória principal de armazenamento etc. Este nível de modularização foi definido por Bell (2) e é chamado nível PMS (Processor Memory Switch).

O outro nível de modularização apresenta um conjunto de módulos padrão com um número maior de elementos. Cada elemento deste conjunto apresenta-se com complexidade e potencialidade bem menor que os do nível PMS. Estes módulos são basicamente as peças que compõem as unidades funcionais dos blocos PMS. Este nível também foi definido por Bell (2) como sendo o nível RT (Register Transfer). Módulos típicos de um sistema modularizado em nível RT são : registradores,unidades de armazenamento de pequena capacidade, unidades lógicas e aritméticas, unidades de seleção, módulos de controle, etc...

Um sistema digital pode apresentar modularização PMS, possuindo blocos do tipo processador central, canais de entrada e saída de dados, memória principal e ainda possuir modularização em nível RT dentro de cada um dos módulos PMS.

É claro, a esta altura, que não são estes os dois únicos níveis de modularização possíveis de serem definidos. Na literatura especializada (1) é suposta a existência de um terceiro nível, que ficaria situada entre o PMS e o RT. Este terceiro nível não está claramente definido; portanto, atualmente a classificação dos sistemas digitais segundo os níveis de modularização restringe-se somente aos dois níveis apresentados.

Diversos minicomputadores modernos são modulares em nível PMS: Burroughs D825(1), MODCOMP série, GRI-909 (3)Computer Corporation, PDP-11 Digital Equipment Corporation e muitos outros com organizações parecidas às destes.

Observando a estrutura desses minicomputadores pode-se identificar uma característica comum a todos eles, a qual aparentemente permite a modularização em nível PMS. Os blocos funcionais do tipo processador central, memória principal, unidades controladoras de entrada e saída de dados são conectadas a uma via comum, através da qual os blocos se comunicam obedecendo a um protocolo fixo e previamente definido.

Para que um sistema possua um protocolo fixo para comunicação entre seus blocos é necessário que seja definido um conjunto de sinais de informação, de controle e de sincronismo e uma ou mais sequências fixas de troca destes sinais entre os blocos do sistema através da via comum, as quais chamaremos de diálogos.

Uma vez esclarecidos os principais objetivos a serem alcançados no projeto lógico do minicomputador, definidos os dois níveis de modularização segundo os quais pode-se classificar um computador e identificadas as propriedades comuns a diversos sistemas que possuem modularização em nível PMS, pode-se passar ao estudo mais detalhado do projeto lógico com modularização em nível de transferência entre registradores (RT).

2. Projeto lógico com modularização em nível de transferência entre registradores (RT). Os módulos.

Para que um sistema digital possua modularização em nível RT é preciso que seja composto por unidades lógicas pertencentes ao conjunto dos módulos padrão do nível RT. Estes elementos são dos seguintes tipos:

Registradores, Registradores deslocadores, Registradores Contadores, Unidades de Armazenamento de pequena capacidade, Unidades lógicas e aritméticas, Unidades multiplexadoras (selecionadoras), Unidades decodificadoras, Unidades transcodificadoras, etc...

Mesmo que um sistema digital seja formado por elementos lógicos do tipo dos acima citados, não temos uma condição suficiente para que ele seja modular em nível RT. É necessário que estes elementos estejam interconectados segundo uma sistemática (projeto) fixa, bem definida e suficientemente generalizada para dotar o sistema de uma grande flexibilidade do ponto de vista de alterações nos algorítmos implementados ou mesmo de inclusão de novos algorítmos.

Fazendo-se uma analogia com o projeto de programas em computadores digitais, pode-se dizer que os conceitos utilizados no projeto de um sistema digital modularizado em nível RT são semelhantes, em alguns aspectos, aos da programação estruturada em "software". Ou seja, abre-se mão da otimização em alguns pontos isolados do sistema para se ter uma estrutura mais genérica, mais flexível, de mais fácil entendimento, de maior capacidade e melhor estruturada.

É claro que a adição de tais qualidades num sistema possui também as suas desvantagens. O sistema resultante, dependendo da sua complexidade, pode-se tornar mais caro e algumas vezes mais lento. Todos estes fatores são pontos importantes na determinação da sistemática de projeto lógico a ser usada no projeto de um sistema digital.

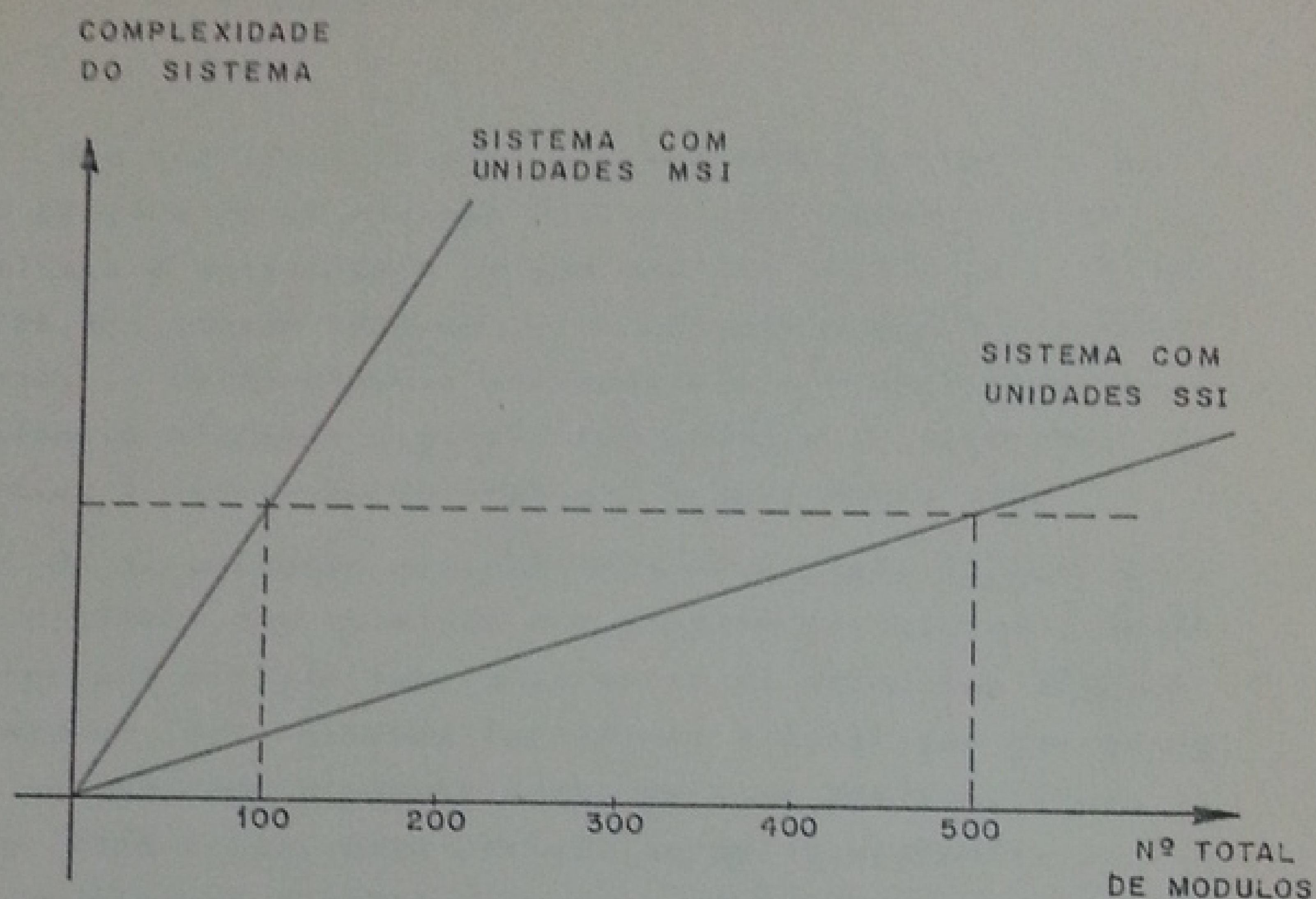
O surgimento dos circuitos integrados em média escala tornou viável a implementação de sistemas digitais, do porte de pequenos minicomputadores com técnicas de projeto estruturado. (Daqui para frente iremos confundir os conceitos de projeto lógico estruturado e projeto lógico modularizado em nível RT). Os circuitos integrados em média escala (MSI) disponíveis atualmente formam um conjunto onde encontramos diver

sas unidades lógicas padrão para um sistema modular em nível RT, e com a grande conveniência de que seus preços estão sendo reduzidos dia a dia. Sistemas que há alguns anos eram economicamente inviáveis de se implementar segundo as sistemáticas de projeto estruturado, devido ao elevado preço das unidades MSI ou mesmo pela não existência das mesmas, hoje podem ser construídos. Eles acabaram resultando em sistemas fisicamente bem menores e algumas vezes até mais baratos que os seus equivalentes construídos com circuitos integrados em pequena escala (SSI).

A redução de tamanho do sistema se deve à grande compactação conseguida com os circuitos integrados MSI. Uma unidade MSI possui uma capacidade muito maior que uma unidade SSI, para um mesmo empacotamento (tamanho físico). O número de portas lógicas que compõem um circuito integrado MSI típico varia de 20 a 100 (14). Portanto para uma mesma capacidade global do sistema necessita-se de um menor número de unidades MSI que SSI para realizá-lo.

Com a redução de preço das unidades MSI, pode-se chegar até a uma redução no preço do sistema, fato este que pode ser observado nas listas de preços de minicomputadores no decorrer dos anos 1972 e 1973.

Tentando relacionar a capacidade global de um sistema digital e o número de módulos MSI ou SSI que seriam necessários para realizá-lo, pode-se empiricamente traçar o gráfico ao lado:



Para ilustrar melhor o que se tentou explicar, vamos recorrer a um exemplo real(4). Um mesmo sistema (computador M6800 Motorola), foi implementado primeiramente usando-se somente unidades SSI, onde foram necessárias 5 placas de circuitos impresso, sendo utilizados cerca de 500 módulos SSI. Já a implementação com unidades MSI necessitou somente 1 placa do mesmo tamanho com cerca de 100 circuitos integrados.

3. Sistemáticas de projeto lógico

Uma vez identificadas as condições necessárias para se ter o projeto de um sistema digital estruturado, tendo sido ressaltada a necessidade de uma análise prévia de certos parâmetros que possam compromoter a relação preço/performance do sistema, e evidenciada a conveniência e viabilidade de se ter atualmente sistemas digitais com projeto estruturado, podemos passar à discussão do problema propriamente dito.

Ao se procurar sistemáticas de projeto lógico para sistemas digitais que resultem num projeto estruturado, convém estabelecer uma divisão funcional entre os circuitos lógicos que os compõem. Esta divisão facilitará a definição dos procedimentos necessários e convenientes para um projeto lógico estruturado. Além disso, esta divisão surge naturalmente ao se analisar um sistema digital, tornando mais claro e simples o seu entendimento.

Segundo Husson[5], um sistema digital pode ser descrito de uma maneira simples, como sendo uma rede de circuitos lógicos chamada fluxo de dados. As unidades lógicas que compõem o fluxo de dados, são circuitos que realizam operações sobre os dados manipulados pelo sistema. Estas funções podem ser encaradas como sendo operações booleanas estáticas do tipo: adição, subtração, deslocamento, geração de paridade, armazenamento, funções arbitradoras de prioridades, seleção e outras.

Uma função estática é uma operação especifíca realizada por uma entidade lógica do fluxo de dados, em um ou mais ciclos de máquina. Estas entidades funcionais podem ser compostas por circuitos lógicos sequenciais ou combinacionais, projetados para armazenar ou realizar uma operação booleana num determinado dado.

Estas unidades lógicas estáticas são interconectadas através de vias de dados, as quais permitem que as informações fluam de uma unidade funcional para outra e sejam armazenadas ou transformadas.

Todas estas unidades são de natureza estática e podem ser comandadas por sinais elétricos de ativação ou desativação, que emanam dos circuitos de controle. Os sinais de controle podem ser pulsos extraídos de uma base de tempo interna do sistema e que servem para ordenar, no tempo, as funções a serem realizadas. O controle ainda é responsável por sinais de decodificação, sinais de sequencialização de operações e sinais de decisão lógica. Estes sinais dirigem e controlam a operação do sistema por alguns ciclos consecutivos.

A unidade funcional de controle pode ser descrita como um circuito lógico combinacional e sequencial. A parte combinacional realiza as operações sobre os campos de dados. A parte sequencial coloca o sistema no estado correto para realizar a próxima operação necessária à execução do comando dado ao sistema digital.

Por exemplo, num sistema de processamento de dados, a lógica sequencial do controle busca a instrução de máquina da memória principal e a decodifica para determinar que sequencia deve ser seguida. Esta sequencia é sincronizada por pulsos de tempo e modificada pelas ordens de desvios emitidas pelo controle. A sequência determina quais os circuitos lógicos que devem ser ativados no fluxo de dados, quais os testes que devem ser realizados e qual a sequencia seguinte a ser seguida. Os pulsos de tempo determinam quando os circuitos lógicos selecionados no fluxo de dados devem ser ativados e por quanto tempo eles permanecerão ativados. Os testes e as decisões lógicas determinam o estado da máquina e ativam a lógica de controle condicional, que é usada para selecionar entre duas ou mais sequências de controle alternativas.

Através da análise funcional de um sistema digital, conclue-se que ele pode ser dividido em duas partes distintas: Fluxo de dados e Controle. Como foi visto na descrição feita acima, pôde-se identificar claramente estas duas partes, ressaltando inclusive algumas de suas funções específicas. Esta divisão facilitará muito a nossa pesquisa para se obter uma sistematização do projeto lógico estruturado de um sistema digital, Devido às diferenças existentes entre as funções realizadas pelas unidades lógicas do fluxo de dados e do controle, iremos também dividir a nossa procura em duas partes. Será estabelecido um procedimento para o projeto do fluxo de dados e outro para o projeto do controle.

Antes de iniciar o estudo para a obtenção das sistemáticas, deve-se lembrar que atualmente existem duas alternativas de ataque deste problema, segundo o conjunto de módulos padrão RT a ser utilizado. Uma delas é a de criar uma sistemática extremamente simples e objetiva juntamente com a definição de um conjunto de módulos padrão, não tomando como vínculo os circuitos MSI já existentes. Esta alternativa é considerada por muitos pesquisadores na área [6,7,8], que têm como objetivo também a definição de novos módulos no conjunto dos circuitos integrados MSI. A segunda alternativa é a de criar sistemáticas de projeto lógico estruturado tentando utilizar os módulos RT disponíveis atualmente [9,10].

Dentro do contexto deste trabalho, tendo em mente os principais objetivos de projeto, deveremos nos concentrar na segunda alternativa, ou seja, a sistematização do projeto lógico utilizando módulos RT disponíveis através do conjunto de circuitos integrados MSI.

4. Projeto do fluxo de dados

Identificadas as duas partes que compõem um sistema digital, pode-se perceber que devido às suas características particulares, ambas requerem módulos de características diferentes para o seu projeto.

O fluxo de dados de um sistema digital é constituído basicamente por unidades de armazenamento, chamadas registradores, que guardam informações codificadas na forma de um conjunto de bits. Os registradores são conectados através de vias de transferência, as quais permitem que a informação contida em um registrador seja transferida para outro através de unidades funcionais estáticas. Estas por sua vez aceitam como entrada um conjunto de bits e fornecem como saída alguma função booleana destes bits.

Elementos extremamente úteis no projeto do fluxo de dados de um sistema digital são: registradores (74175,74174), registradores deslocadores (74194,74198), unidades lógicas e aritméticas (74181,74281),registradores contadores (9316,74193), unidades de armazenamento de pequena capacidade (rede de registradores) (3101, 93403) e multiplexadores (74158,9309). Todos estes módulos pertencem ao conjunto dos circuitos integrados MSI TTL (transistor transistor logic) disponíveis atualmente. Os números colocados entre parênteses correspondem à identificação dada pelo fabricante a essas unidades lógicas.

Para projetar o fluxo de dados de um sistema digital é preciso conhecer perfeitamente os algorítmos a serem executados e o conjunto de módulos RT disponíveis.

Da análise dos algorítmos que deverão ser executados pelo sistema, identificam-se as unidades funcionais estáticas que necessariamente ou preferivelmente deverão participar do fluxo de dados. Os algorítmos que o sistema deverá executar são definidos quando se estabelece a arquitetura do sistema em projeto.

Entende-se por arquitetura de um sistema digital, o seu comportamento do ponto de visto do usuário. Quando se define o que o sistema deve fazer para o usuário, deve-se definir também os algorítmos realizadores das suas diversas funções. (Pelo menos numa primeira aproximação).

A identificação e escolha das unidades funcionais estáticas, que irão compor o fluxo de dados, não é suficiente para a sua definição total. O ponto mais importante no estabelecimento do fluxo de dados de um sistema digital são as interligações que deverão existir entre as unidades funcionais estáticas já escolhidas. (Topologia do Fluxo de Dados).

A determinação dos caminhos de informação (vias)que deverão existir no fluxo de dados, também está fortemente ligada aos algorítmos realizadores das funções do sistema.Analisando-se as sequências de operações sugeridas pelos algorítmos e observando as operações que se supõe serem realizadas simultaneamente, pode-se identificar as vias principais, mais apropriadas para formar o fluxo de dados. É claro, que as soluções para este problema são infinitas, mas devido à extrema simplicidade do mesmo, é fácil chegar rapidamente a uma solução. Através do exemplo a seguir pode-se perceber a facilidade em se obter soluções para este problema.

Associado ao projeto do fluxo de dados tem-se também um outro fator importantíssimo que é o tempo de realização das operações pelas unidades funcionais estáticas. Estabelecido o fluxo de dados de um sistema digital, pode-se definir o seu ciclo. O ciclo do fluxo de dados é o tempo que um conjunto de bits leva para sair de uma unidade funcional estática e se transferir para outra, sofrendo transformações ou não, percorrendo o caminho mais demorado existente no fluxo de dados. O ciclo do fluxo de dados, em geral, é usado como o tempo básico para a realização de uma operação no fluxo de dados, ou seja, transferir informação de uma unidade para outra independentemente da via utilizada.

Desta forma o ciclo do fluxo de dados é um parâmetro muito importante, quando deseja-se ter um sistema rápido, ou mesmo quando a análise prévia dos algorítmos nos leva a várias soluções, aparentemente equivalentes do ponto de vista do número de unidades funcionais estáticas utilizadas e da rede de vias de informação que as interligam. Nestes casos recomenda-se escolher a solução que apresentar o menor ciclo.

Em alguns casos, quando o objetivo é a obtenção de um sistema bastante rápido, para poder diminuir o ciclo do fluxo de dados é necessário criar mais vias de informação interligando as unidades funcionais estáticas. Estas vias secundárias servem como rotas alternativas para transferência de informação, no caso de operações simultâneas realizadas no fluxo de dados. Quando existem muitas vias secundárias, diz-se que o fluxo de dados possui um alto grau de paralelismo, fato este que permite a realização de um determinado conjunto de operações num número menor de ciclos do fluxo de dados. Desta forma obtém-se um sistema mais veloz.

É conveniente ressaltar que, quando estiver sendo analisado o grau de paralelismo necessário para se implementar um determinado algorítmo, pode ser conveniente alterar o próprio algorítmo, diminuindo o número de operações simultâneas em benefício de um fluxo de dados com um menor número de unidades e uma rede de interligação não tão volumosa.

Uma outra maneira de se obter um ciclo menor para o fluxo de dados, é escolher módulos RT mais velozes. Uma unidade funcional estática possui um atraso inerente aos circuitos lógicos que a compõe. Este atraso é o tempo entre o instante em que são fornecidas as entradas até o instante em que as saídas tenham o resultado da operação. Uma mesma unidade funcional pode-se apresentar em várias versões que diferem somente nos atrasos associados às operações por ela realizada. Na prática, pode-se ter circuitos integrados MSI da família TTL com velocidade normal (10ns por "gate")+, alta velocidade (6ns por "gate") - super alta velocidade (3ns por "gate").

Para sistemas com o mesmo grau de paralelismo, utilizando-se módulos com menor atraso, diminui-se o ciclo do fluxo de dados tornando o sistema mais veloz.

Portanto, pode-se observar a existência de um compromisso entre o grau de paralelismo, o ciclo e o tamanho do fluxo de dados. Os objetivos do projeto é que determinam para qual dos três pontos deve se dar maior ênfase.

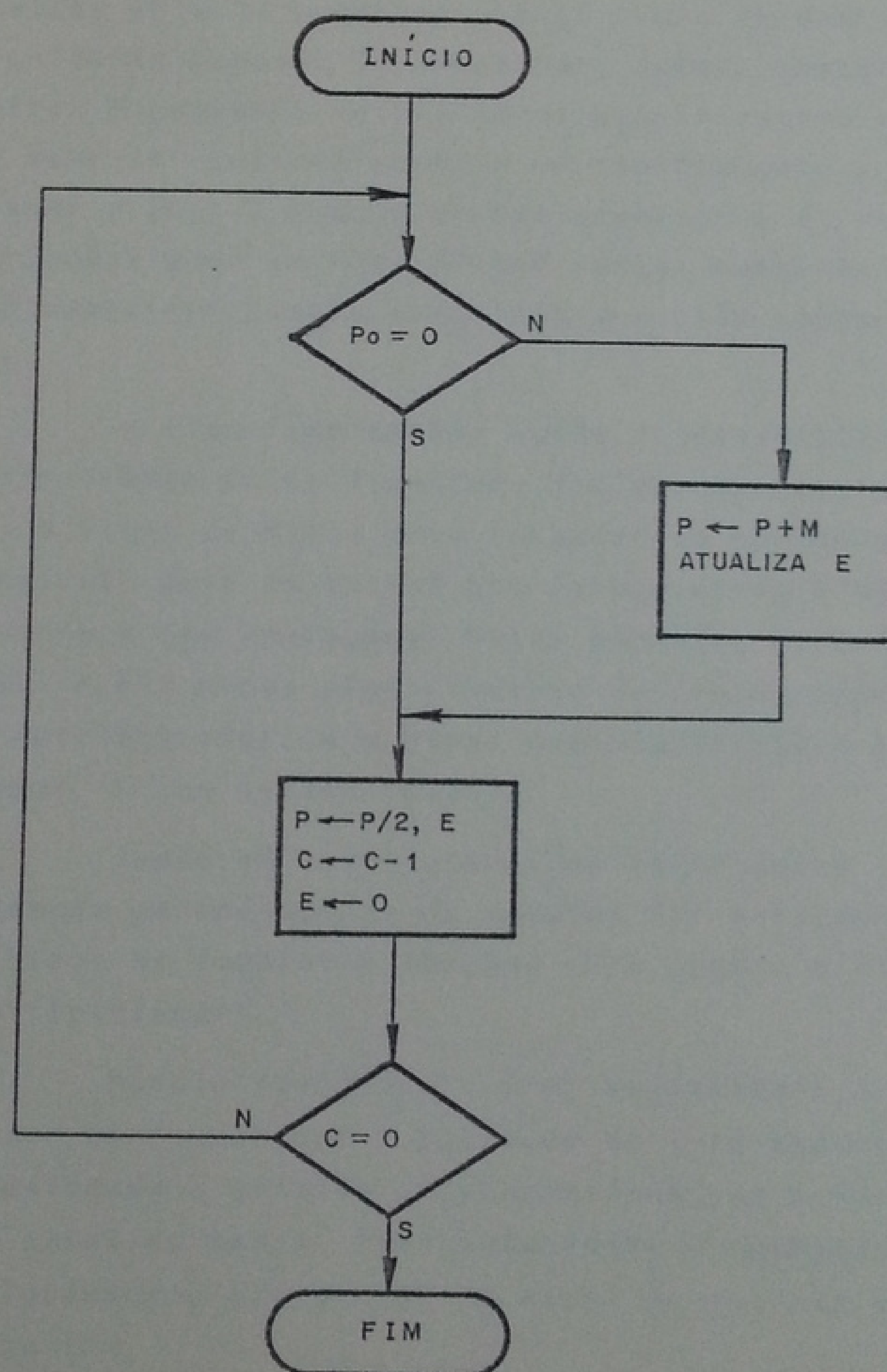
Para poder escolher os módulos RT mais convenientes para compor o fluxo de dados, é necessário um perfeito conhecimento dos circuitos integrados MSI disponíveis. Não só é necessário conhecer todos os módulos, mas também conhecer cada um deles em todos os seus detalhes, pois só assim tem-se condições para identificar os módulos RT mais apropriados para o fluxo de dados, e que possam até facilitar as funções do controle. A descrição dos circuitos integrados MSI aparecem em manuais fornecidos pelos fabricantes, contendo todas as informações técnicas necessárias.

Antes de passar a um exemplo de projeto de fluxo de dados, convém resumir o procedimento recomendado para projetá-lo.

O projetista deve ter um conhecimento total e perfeito dos módulos RT que estão a sua disposição. Os algorítmos que foram estabelecidos na definição da arquitetura do sistema digital em projeto devem ser analisados para se identificar as unidades funcionais estáticas que irão compor o fluxo de dados. Através da sequencia de operações sugeridas pelos algorítmos pode-se identificar uma ou mais maneiras de interligar as unidades funcionais estáticas, criando assim as vias de informação (definição da topologia). Para a distinção entre as várias soluções propostas, usa-se o valor do ciclo do fluxo de dados como critério de seleção, lembrando sempre os objetivos do projeto para poder chegar a uma solução de compromisso entre o grau de paralelismo desejado, o ciclo e o tamanho do fluxo de dados.

Exemplo 1: Projetar o fluxo de dados de um sistema digital que realiza a multiplicação de dois números binários inteiros e positivos de 4 bits, resultando num produto de 8 bits. Assumir que o algorítmo usado deve ser o de deslocamentos à direita.

Algorítmo:



Condições iniciais: C = 4; M = multiplicando P 0:3 = multiplicador; P 4:7 = Produto parcial = 0; E = extensão = 0

Condições finais: C = 0; M = multiplicando; P 0:7 = produto.

Analisando as funções necessárias para a execução do algorítmo pode-se perceber que no fluxo de dados devem existir unidades capazes de armazenar, somar, contar e deslocar à direita. Dependendo do bit menos significativo do multiplicador, deve-se realizar somente um deslocamento à direita ou uma soma e logo a seguir um deslocamento à direita. Com estas informações pode-se identificar várias maneiras de interligar as unidades funcionais estáticas que irão compor o fluxo de dados.

O algorítmo também supõe a possibilidade de operação simultânea entre a unidade deslocadora e a unidade contadora. O fluxo de dados deve fornecer ao controle as variáveis necessárias para os testes que determinarão a sequencialização correta das operações. Neste exemplo, as variáveis de teste são; o bit menos significativo do registrador que armazenará o multiplicador, e o sinal que identifica a presença do contador C no estado $(0000)_2$.

Tendo em mente a análise feita sobre o algorítmo e conhecendo perfeitamente os módulos MSI existentes, podem ser escolhidos os seguintes módulos para compor o fluxo de dados do multiplicador:

Módulo deslocador: É um registrador de 4 bits que pode ser carregado com o conteúdo de suas quatro entradas ou ser deslocado à direita em sincronismo com a borda de subida de um sinal de tempo. Para selecionar a operação paralela ou o deslocamento, ele possui um sinal de entrada específico para esse fim.

No caso das operações de deslocamento existe ainda um sinal de entrada que serve para posicionar o bit que vai ser inserido no registrador. O conteúdo deste registrador pode ser feito igual a $(0000)_2$ assincronamente através da ação de um outro sinal de entrada. Ele possui como saída os seus quatro bits. Este módulo pode ser identificado pelo número 9300 no catálago Fairchild-TTL.

Módulo registrador: registrador de 4 bits que pode ser carregado com o conteúdo de suas entradas em correspondência com a borda de subida do sinal de tempo, ou o seu conteúdo pode ser feito assincronamente (em relação ao sinal de tempo) igual a zero através de um outro sinal de entrada. Este módulo pode ser identificado pelo número 74175 no catálogo da Signetics-TTL.

Módulo somador: realiza a soma de dois números binários colocados em suas respectivas entradas. Possui uma entrada de "vem-um" que permite a sua conexão com outros módulos semelhantes para formar unidades somadoras com mais de 4 bits. Além de fornecer o resultado ele fornece também uma saída de "vai-um" (extensão). Este módulo pode ser identificado pelo número 7483 no catálogo Fairchild-TTL.

Módulo contador: é um registrador de 4 bits que pode ser carregado assíncronamente através de suas quatro entradas paralelas, ou pode incrementar ou decrementar de uma unidade o seu conteúdo em sincronismo com sinais de tempo, um para incrementar e outro para decrementar. Além de fornecer como saída o seu conteúdo, existe um sinal especial de saída que indica a presença do contador no estado $(0000)_2$. Este módulo pode ser identificado pelo número 74193 no catálogo da Signetics-TTL.

Módulo armazenador: é uma unidade capaz de armazenar um bit de informação colocado em sua entrada, em correspondência com a borda de subida do sinal de tempo. Este é um circuito lógico bi-estável do tipo D. Ainda seu conteúdo pode ser feito igual a zero ou a um, através de comando por sinais de entrada que agem assincronamente em relação ao sinal de tempo. Como saída ele apresenta o valor de seu conteúdo e o seu complemento. Este módulo pode ser identificado pelo número 7474 no catálogo da Fairchil-TTL.

São estes os módulos necessários para o fluxo de dados do multiplicador. Uma maneira eficiente de interligá-los, lembrando a ordenação entre as operações requisitadas pelo algorítmo, é a da Figura II.1.

Para finalizar este tópico, devemos ressaltar que os módulos RT disponíveis através do conjunto dos circuitos integrados MSI são bastante eficientes para o projeto do fluxo de dados da maioria dos sistemas digitais de pequeno e médio porte. Devido a este fato e a extrema regularidade que o fluxo de dados apresenta não foi necessário desenvovler uma sistemática mais elaborada que esta que foi aqui apresentada.

5. Projeto do Controle

Para poder detalhar as sistemáticas de projeto lógico dos circuitos que implementam a unidade de controle de um sistema digital, tendo como objetivo principal a obtenção de um projeto lógico estruturado, deve ser esclarecido o que vem a ser este projeto sob um ponto de vista básico e de caráter geral.

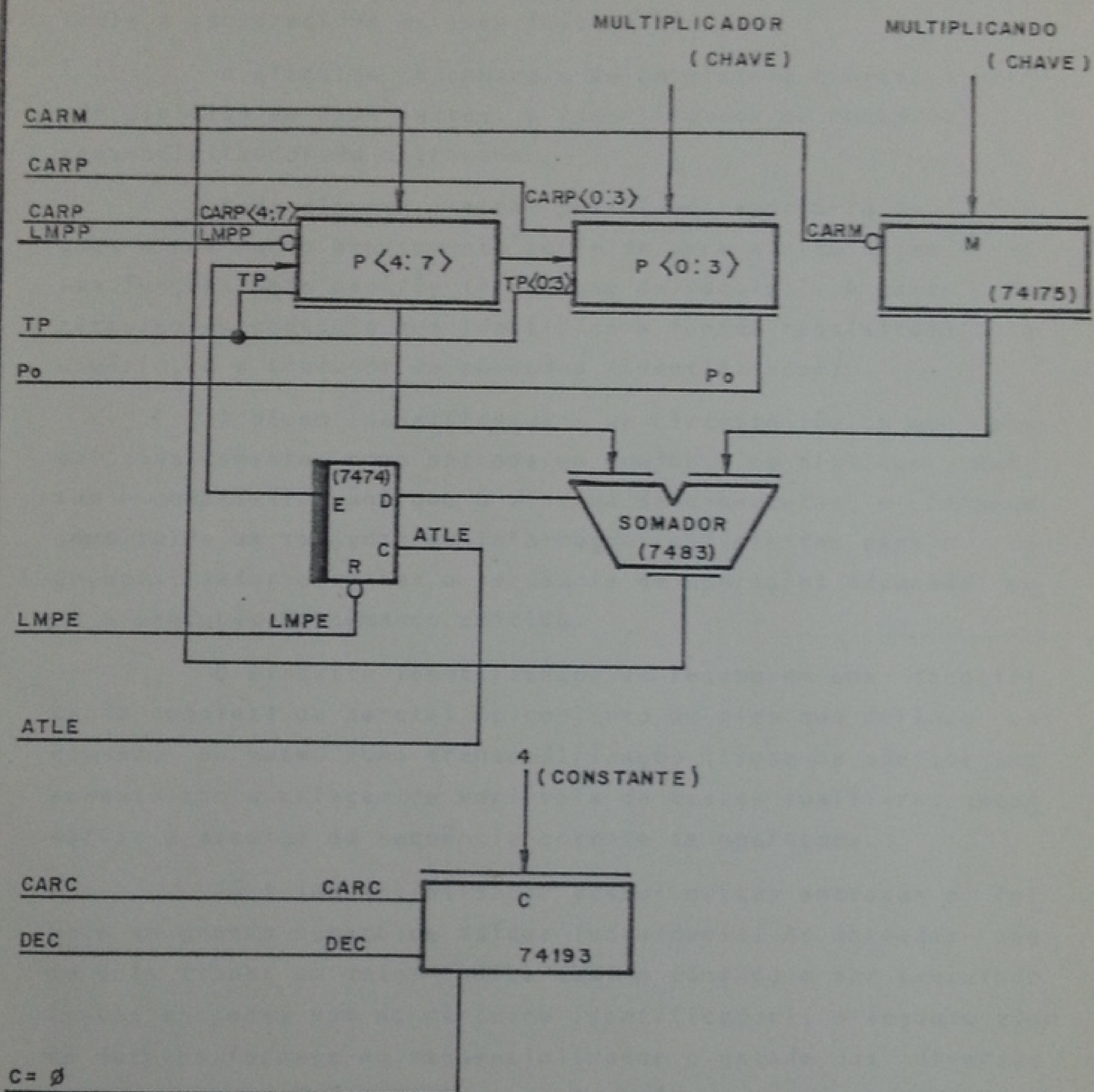


Fig.II.1. - Fluxo de dados do multiplicador

Um enfoque bastante fundamental do que vem a ser o controle de um sistema digital é apresentado por Gschwind[11], onde são identificadas as duas partes de uma unidade de controle e esclarecidos as suas funções básicas.

O circuito de controle de um sistema digital pode ser dividido em duas partes: o identificador de comandos e o sequencializador de operações.

Todo sistema digital possui um repertório de funções e uma regra previamente definida para a requisição dessas funções pelo usuário (linguagem de máquina). A parte do circuito do controle que identifica a função requisitada pelo usuário, é o tradutor de comandos (identificador).

O bloco identificador é um circuito lógico que possui, basicamente, como entrada um conjunto de bits que definem o comando(função) que o sistema deve executar, e fornece como saída um conjunto de informações suficientes para o sequencializador escolher a sequência de operações adequada para a execução do comando emitido.

O circuito identificador se resume em uma decodificação completa ou parcial do conjunto de bits que definem os comandos ou mesmo numa transcodificação (troca de código) juntamente com a criação de variáveis de testes auxiliares necessárias à escolha da sequência correta de operações.

Já o sequencializador possui muitas entradas e fornece um grande número de saídas individuais. As entradas são de dois tipos: um deles indica qual o comando a ser executado (estas entradas vem do circuito identificador); o segunto tipo de entrada fornece ao sequencializador o estado das diversas unidades que compõem o sistema. As saída do sequencializador são sinais de controle que vão agir diretamente sobre os circuitos do fluxo de dados, ordenando ou o armazenamento de informações ou a atualização de bits indicadores de estado.

Enfim, são emitidos um conjunto de comandos de controle necessários para a execução das diversas funções do sistema.

Todos estes comandos emitidos pelo sequencializador têm uma duração determinada e são espaçados convenientemente, levando-se em conta o tempo de execução das diversas operações executadas pelas unidades funcionais estáticas do fluxo de dados. O sequencializador escolhe os espaços de tempo suficientes, baseando-se nas suas entradas indicadoras do estado das diversas unidades do sistema.

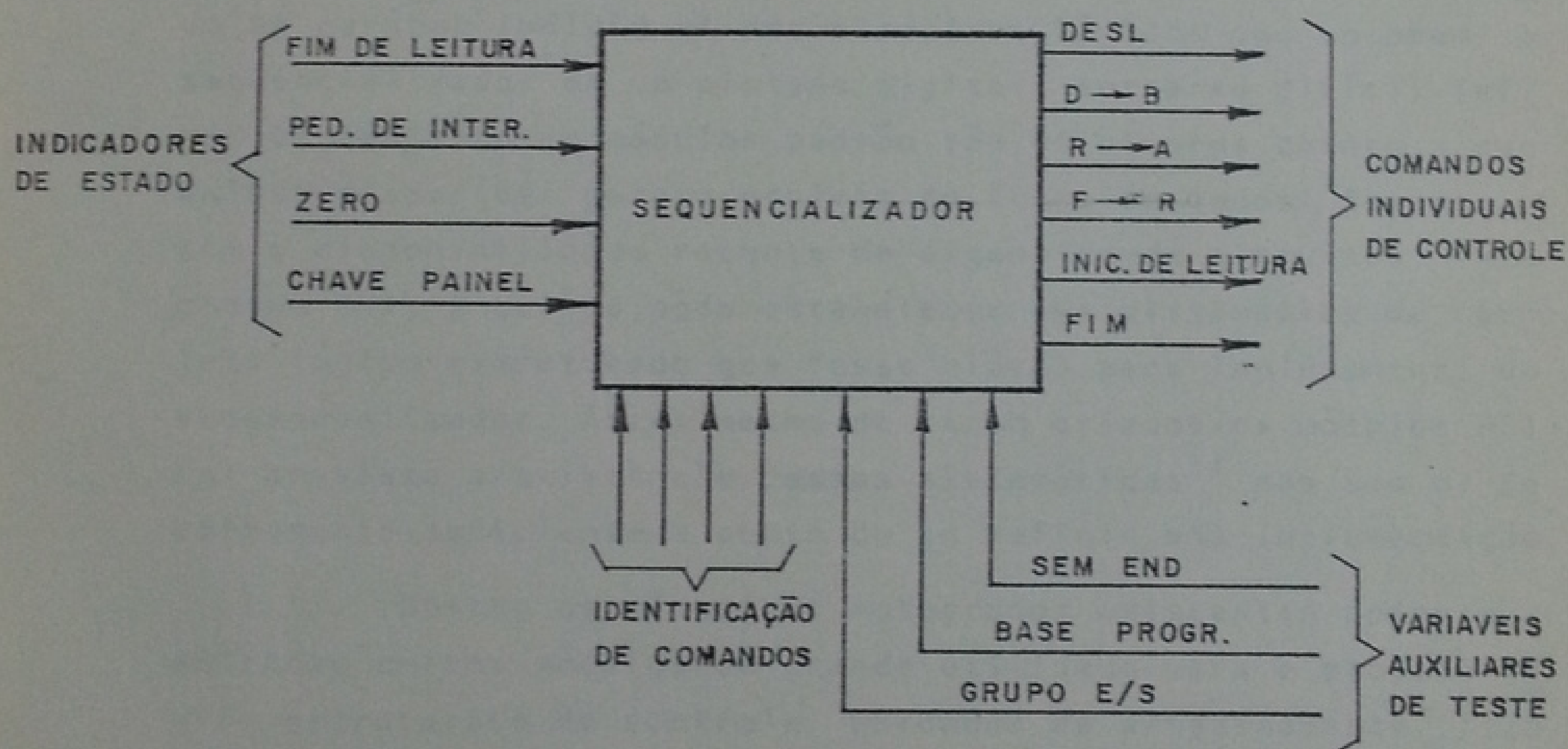


Fig.II.2. - Esquema funcional simplificado de um sequencializador.

Portanto, o problema fundamental do projeto do controle de um sistema digital se resume na obtenção do circuito sequencializador. O circuito identificador, para a maioria dos sistemas digitais, é extremamente simples, resumindo-se quase sempre numa decodificação total ou parcial ou numa transcodificação.

O que se pretende apresentar neste ítem são algumas sistemáticas de projeto de um sequencializador que utilizam eficientemente os circuitos MSI disponíveis, que produzam um projeto simples e de fácil entendimento e que resultem num esquema flexível e estruturado de projeto lógico, facilitando a sua documentação e o próprio teste.

Antes de iniciar a apresentação dos procedimentos de projeto lógico, convém comentar alguns aspectos sobre os módulos RT (MSI disponíveis) para o projeto do controle. Devido ao caráter individual dos circuitos lógicos que compõem o sequencializador de um sistema digital, torna-se difícil definir um conjunto de módulos padrão tão eficientes quanto o que existe disponível para o projeto do fluxo de dados. Somente com a disponibilidade recente de alguns novos circuitos integrados MSI, é que se pôde estabelecer uma sistemática de projeto lógico estruturado que fosse viável para implementar um sequencializador. Antes mesmo de serem criados os módulos MSI foi prevista a existência dessas sistemáticas[11] mas sem o detalhamento suficiente a ponto de se definir uma implementação.

Dentre os circuitos integrados existentes pode-se destacar certos módulos de grande utilidade para o projeto lógico estruturado do controle: unidades de armazenamento fixo (ROM), rede lógica progrâmavel (PLA), unidades decodificadoras, unidades selecionadoras, unidades contadoras, registradores, registradores deslocadores e unidades de armazenamento.

As unidades de armazenamento fixo (ROM) e as redes lógicas programáveis (PLA), dentre os módulos destacados, são os que apresentam maior potencialidade. As unidades de armazenamento fixo são memórias do tipo ler somente que podem ter seu conteúdo gravado uma única vez, não sendo mais destruído. Estas unidades podem ser gravadas pelo fabricante através de uma máscara especificada pelo comprador ou pelo próprio usuário através de um circuito elétrico próprio para tal. Este segundo tipo é chamado memória ler somente programável eletricamente.

No caso da implementação de um protótipo é aconselhável o uso de memórias fixas programáveis eletricamente. Já para uma versão definitiva e destinada a ser reproduzida muitas vezes, é preferível o uso de memórias fixas programáveis através de máscara.

As memórias fixas são de grande eficiência para implementar certas partes do circuito sequencializador, onde se quer condensar num conjunto pequeno de módulos uma grande quantidade de lógica, dando ao circuito uma flexibilidade enorme, fato este que poderá ser observado através dos exemplos dados no capítulo IV, quando será apresentado o sequencializador do processador em projeto.

Por ora, pode-se exemplificar a grande versatilidade de síntese lógica desses módulos escolhendo-se a seguinte unidade : uma memória fixa de 256 bits, organizados em 32 palavras de 8 bits cada uma. Com este módulo pode ser implementada qualquer função booleana com um número de variáveis de entrada menor ou igual a 5 e um número de variaveis de saída menor ou igual a 8. Fato importante é que para alterar a função booleana que foi implementada por outra função das mesmas variáveis, basta programar uma nova memória fixa e substituir a antiga, sem nenhuma mudança de fiação ou "lay-out" nas placas de circuito do sistema.

As redes lógicas programáveis apresentam também um grande poder de síntese lógica combinatória, com a desvantagem de não serem programáveis eletricamente, sendo possível somente a programação pelo fabricante através da máscara definida pelo comprador. Devido a este fato, daqui por diante, não serão feitas mais referências as redes lógicas programáveis.

Para o projeto lógico do circuito identificador, os módulos existentes apresentam-se com uma grande eficiência. Módulos como memórias fixas e decodificadores podem resolver praticamente quase todos os problemas de decodificação ou transcodificação. Devido à simplicidade do circuito identificador e à potencialidade dos módulos citados acima, uma vez definidas as entradas e saídas deste circuito ele está praticamente projetado. Por este fato nós não nos deteremos mais no circuito identificador.

Já para o projeto do sequencializador não acontece o mesmo, sendo necessário o desenvolvimento de alguns procedimentos básicos para a obtenção de um projeto lógico fácil e estruturado do mesmo. É justamente a definição dessas sistemáticas de projeto lógico estruturado do sequencializador da unidade de controle de um sistema digital que procurar-se-á estabelecer a seguir.

5.1 Projeto lógico do controle tentando a utilização de um módulo universal. (controle distribuído).

Um dos objetivos deste trabalho é a identificação de sistemáticas de projeto lógico estruturado, que utilizem eficientemente os circuitos integrados MSI existentes. A sistemática que será apresentada a seguir se baseia na existência de um módulo universal de controle[9], que não pertence ao conjunto dos circuitos integrados disponíveis atualmente.

Este módulo é composto por alguns circuitos lógicos bi- estáveis do tipo JK e algumas portas lógicas.

Apesar do módulo universal de controle não ser disponível comercialmente, é extremamente conveniente e esclarecedor a apresentação da sistemática de projeto lógico de um sequencializador que se baseie nele. O estabelecimento deste procedimento de projeto lógico irá exemplificar o que convencionou-se chamar de projeto lógico estruturado. Por outro lado, a grande simplicidade desde módulo de controle e a sua enorme flexibilidade nos levam a arriscar um palpite de que num futuro breve, uma versão deste módulo talvez, ligeiramente diferente será incorporada ao conjunto de circuitos integrados MSI.

Esta sistemática de projeto se baseia na existência do módulo de controle universal, que interligado convenientemente pode realizar uma vasta gama de algorítmos. Sendo assim, antes de se apresentar o procedimento de projeto lógico estruturado do sequencializador, deve-se descrever o módulo de controle destacando-se as suas propriedades lógicas e topológicas.

- O módulo de controle

O módulo de controle, como foi definido[9], é um circuito sequencial pulsado. A característica síncrona do módulo não impede a realização no fluxo de dados de uma estrutura assíncrona. Poder-se-ia definir o módulo de controle[6] através de um circuito que operasse assíncronamente no modo fundamental, fato este que obrigaria à implementação de uma estrutura assíncrona no fluxo de dados, impossibilitando a utilização dos módulos MSI existentes com ampla liberdade.

A presença de um módulo de controle síncrono facilita a utilização, no fluxo de dados, dos módulos MSI existentes e se necessário ainda se pode ter uma estrutura que permite a realização de operações, através das unidades funcionais estáticas, que durem um número inteiro de ciclos de uma base de tempo pré-fixada.

Como se pode notar, para operar um conjunto de módulos universais de controle, com o intuito de formar o sequencializador da unidade de controle de um sistema digital, deve-se dispor de uma base de tempo. Esta base de tempo pode ser implementada através de um oscilador a cristal, que fornecerá o sinal básico de tempo para coordenar todas as transições de estados no sistema. O projeto do circuito base de tempo é detalhado suficientemente na literatura especializada e não tem nenhuma implicação na sistemática de projeto do sequencializador. Portanto, não nos deteremos mais neste ítem e simplesmente suporemos a existência de um sinal básico de tempo.

Será apresentado a seguir o módulo de controle na sua forma mais simples, para poder evidenciar as suas propriedades e o seu funcionamento.

Na figura II.3.., é mostrado um circuito sequencial síncrono (toda mudança de estado é provocada por uma transição do sinal T) com duas entradas I (início), P (pronto) e duas saídas E(executa), S(seguinte). Quando um sinal é aplicado à entrada I, o módulo é ativado e é iniciada a execução da função associada a este módulo, em correspondência com o sinal T (tempo). O sinal que comanda a execução no fluxo de dados é E. A função pode necessitar de vários períodos de tempo (T) para ser executada. Quando terminar a execução da função deve ser fornecido pela unidade funcional responsável pela execução o sinal P. Com o recebimento do sinal P o módulo é desativado e fornece o sinal S que irá ativar o módulo seguinte. Tanto a ativação como a desativação de um módulo ocorrem em sincronismo com o sinal T.

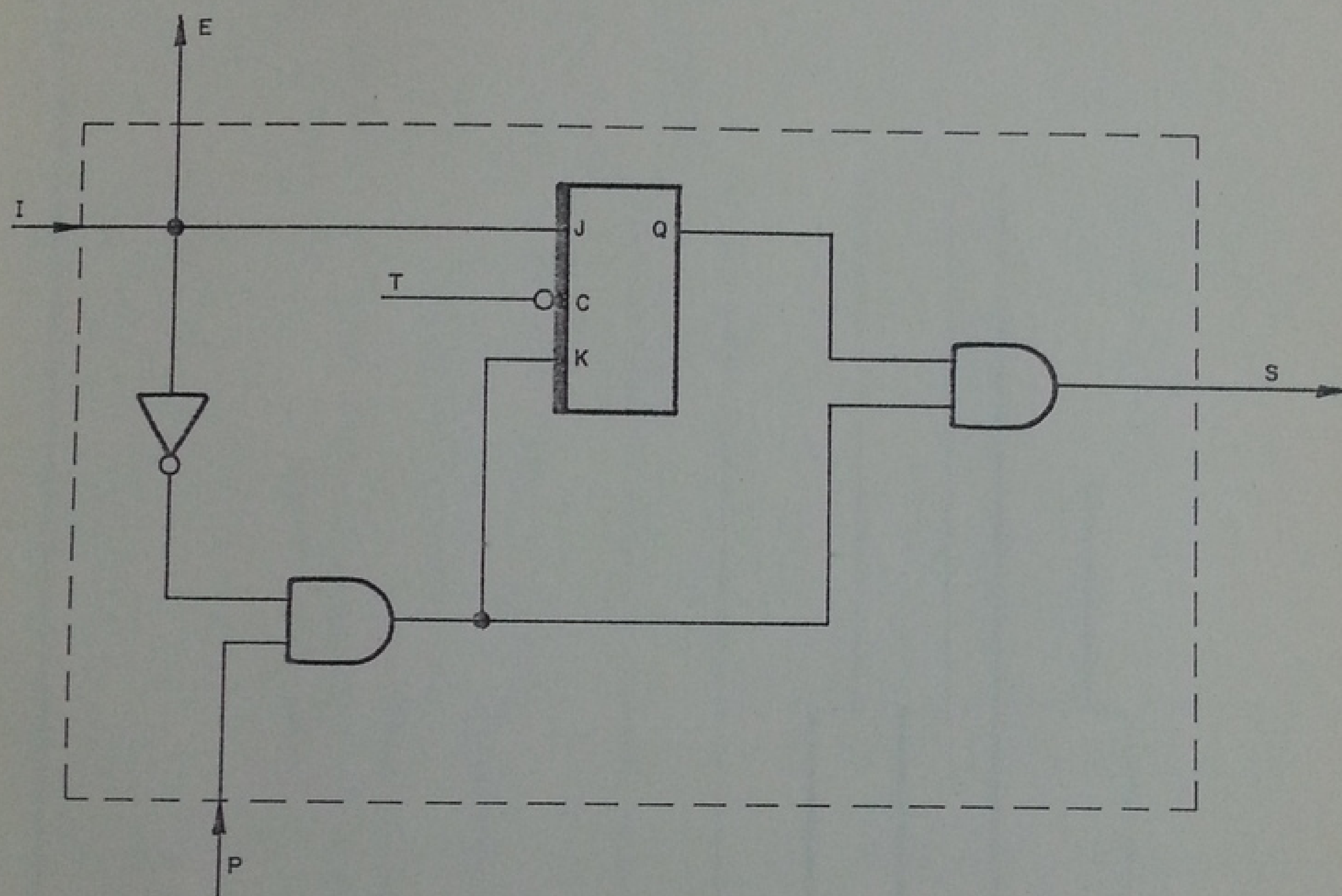


Fig. II.3. - Módulo Universal de controle

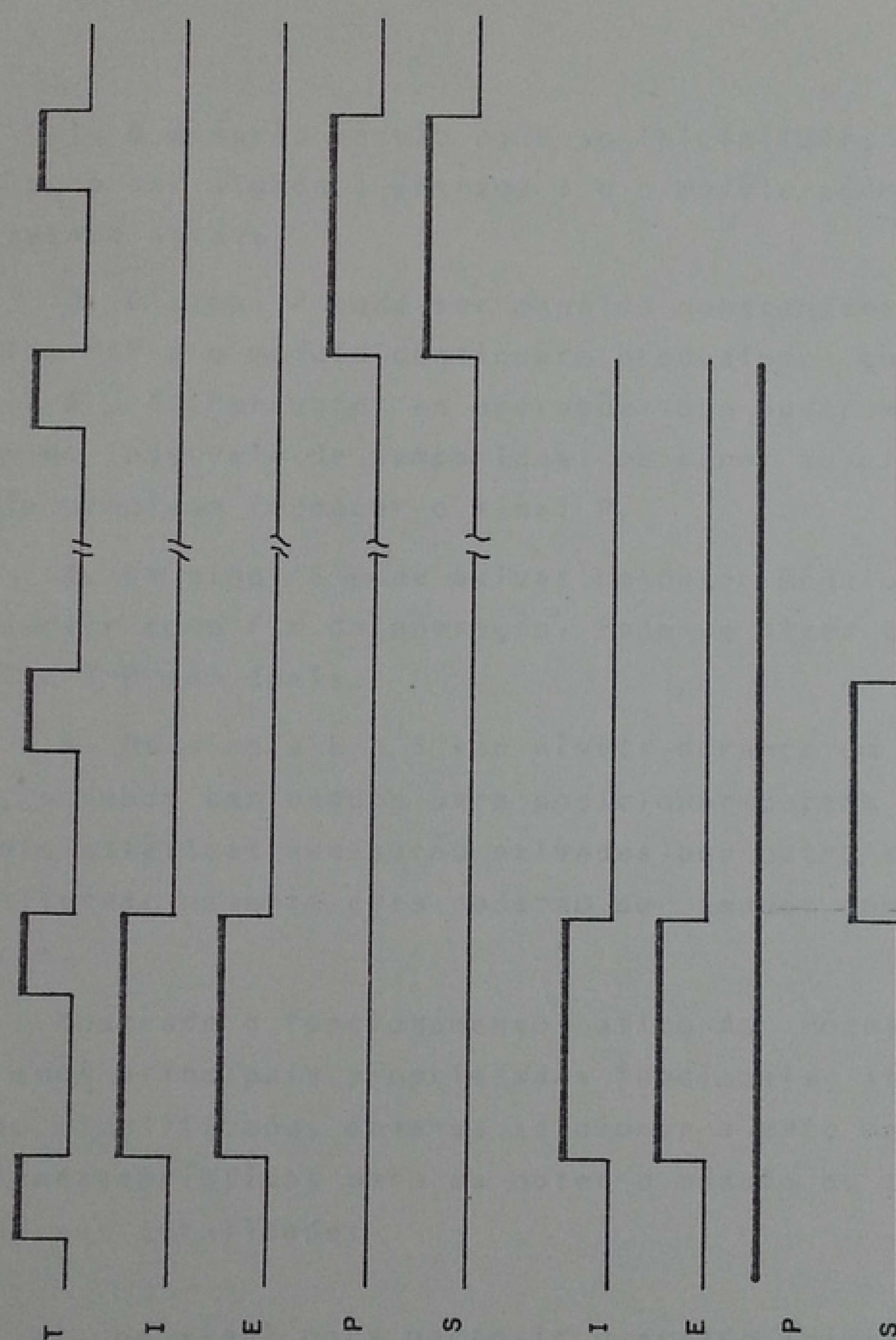


Fig.II.4. - Diagrama de tempo do módulo de controle

- Propriedades do Módulo de Controle

Propriedades funcionais:

1. O próprio módulo pode se inicializar, isto é, a saída S pode ser ligada à entrada I e o módulo permanece ainda num estado estável.

2. O sinal P pode ser mantido constantemente em nível lógico "1" e o módulo continuará produzindo corretamente os sinais E e S. Portanto, as operações que puderem ser realizadas em um intervalo de tempo igual ou menor ao ciclo do sinal T não precisam fornecer o sinal P.

3. Um sinal E pode ativar um outro módulo e o sinal S pode servir como fim de operação. Pode-se dizer que as duplas E-S e I-P são duais.

4. Os sinais E e S são níveis durante um ciclo do sinal T, podendo ser usados para posicionar certas unidades funcionais estáticas que serão ativadas por outro sinal de tempo. Alternativamente eles poderão ser usados como sinais ativadores.

Mostrado o funcionamento básico do módulo e algumas das suas principais propriedades funcionais, através de um modelo simplificado, deve-se adicionar a este modêlo as seguintes características para se obter o módulo de controle universal na sua totalidade:

1. Um sinal para garantir o estado inicial do módulo.

2. Sincronizar a ação do sinal P dentro do módulo com o sinal T, para permitir que o módulo funcione corretamente na execução de funções que durem mais que um ciclo do sinal T.

3. Sinal de entrada que condiciona a ativação do bloco seguinte. Sendo assim, o módulo possuirá duas saídas para ativação S1 e S2. O sinal de condição também deve ser sincronizado com o sinal T, para que o módulo funcione corretamente.

4. Possibilidade de ativação do módulo através de mais de um sinal do tipo I.

Para uma representação esquemática do módulo, adotou-se o seguinte símbolo:

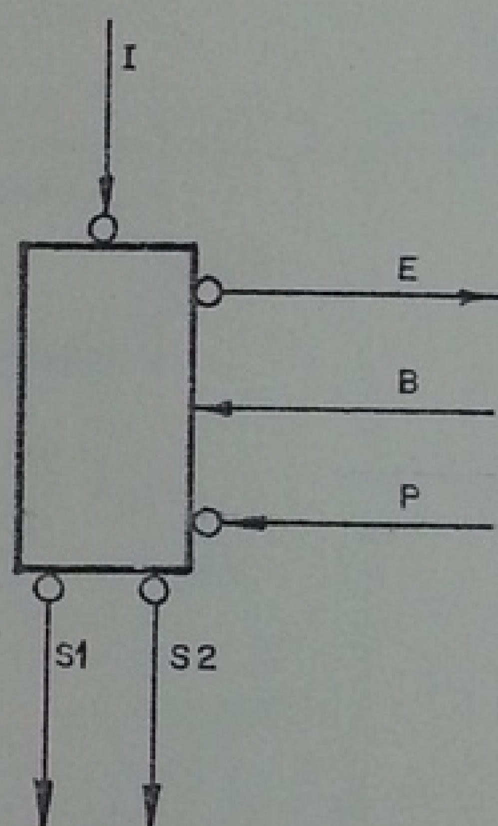


Fig. II.5. Representação simbólica do módulo

Desta forma, o circuito final fica sendo:(representação na figura ao lado).

Dependendo da forma como é utilizado o módulo de controle, ele pode não necessitar de todas as entradas ou saídas que aparecem no circuito completo. Se os sinais não usados forem fixados convenientementes, eles não pertubarão o funcionamento do módulo. Nestes casos, na representação simbólica, deixa-se o módulo simplesmente sem os sinais não utilizados

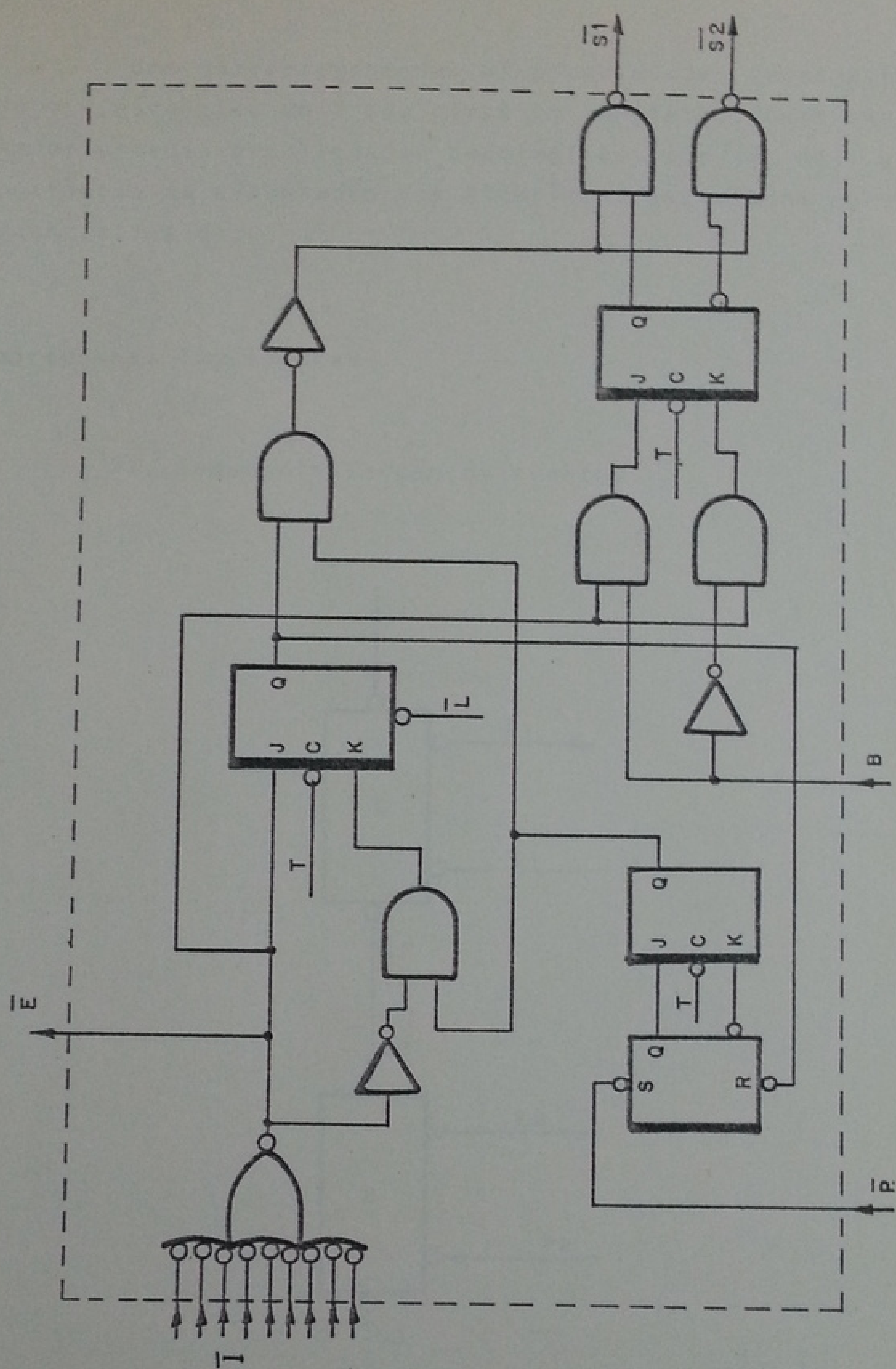
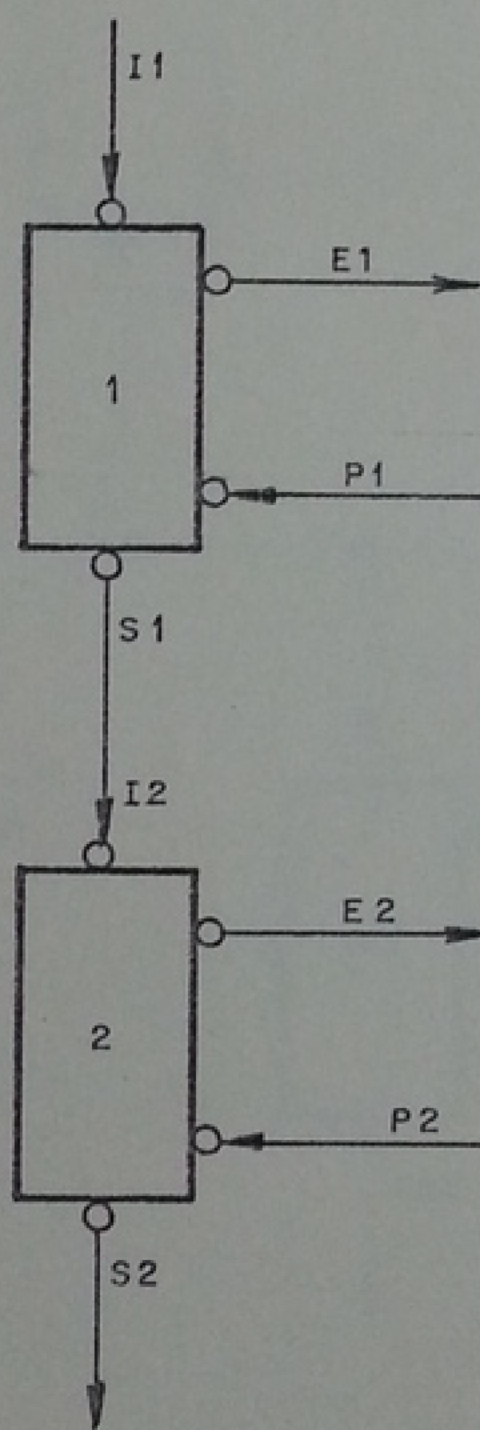


Fig.II.6. - O módulo universal de controle

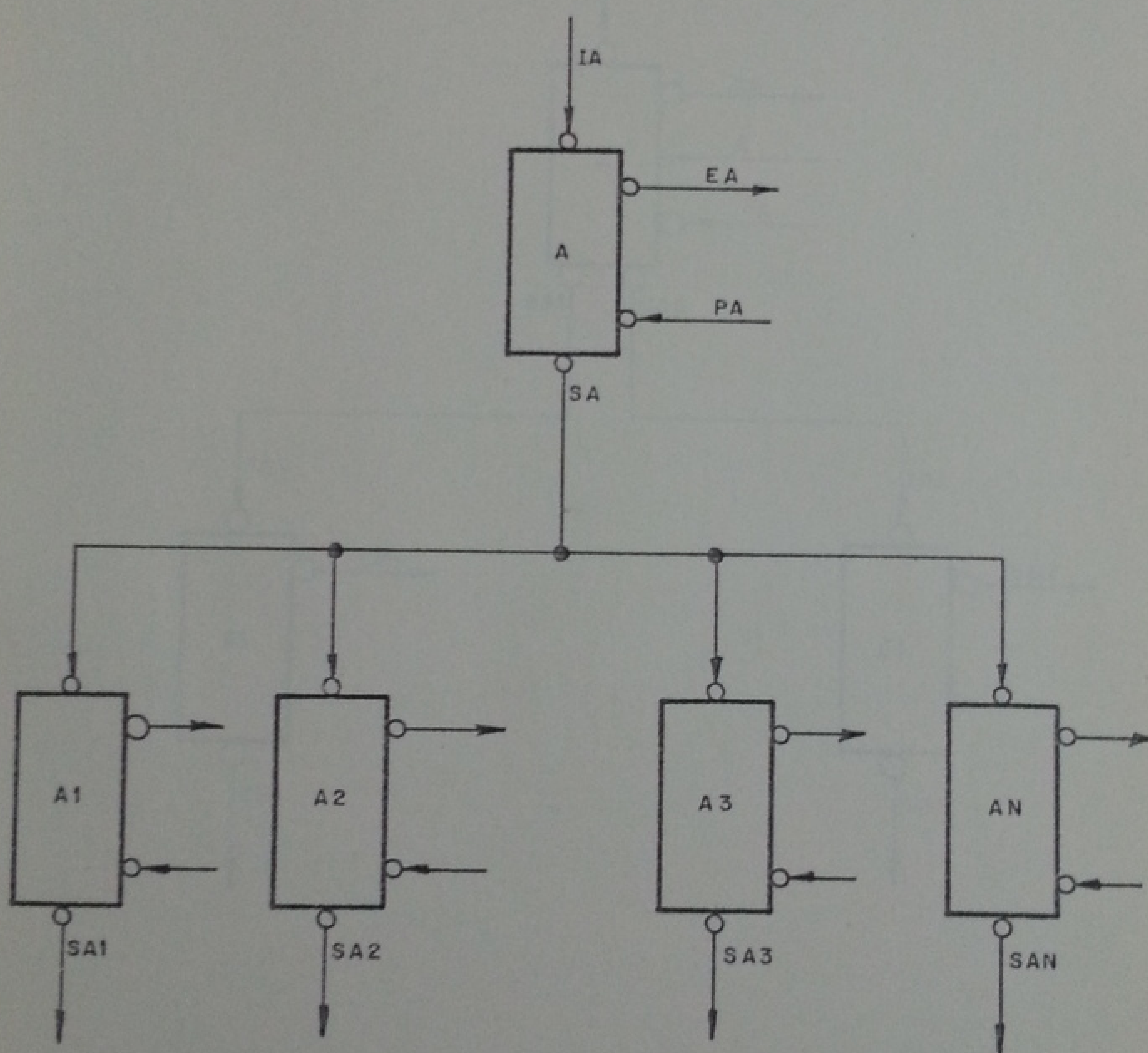
Uma vez apresentadas as propriedades funcionais do módulo e estabelecido o seu circuito completo, devem ser analisadas as suas propriedades topológicas, que são de grande importância na elaboração dos algorítmos executados por um sequencializador.

Propriedades Topológicas:

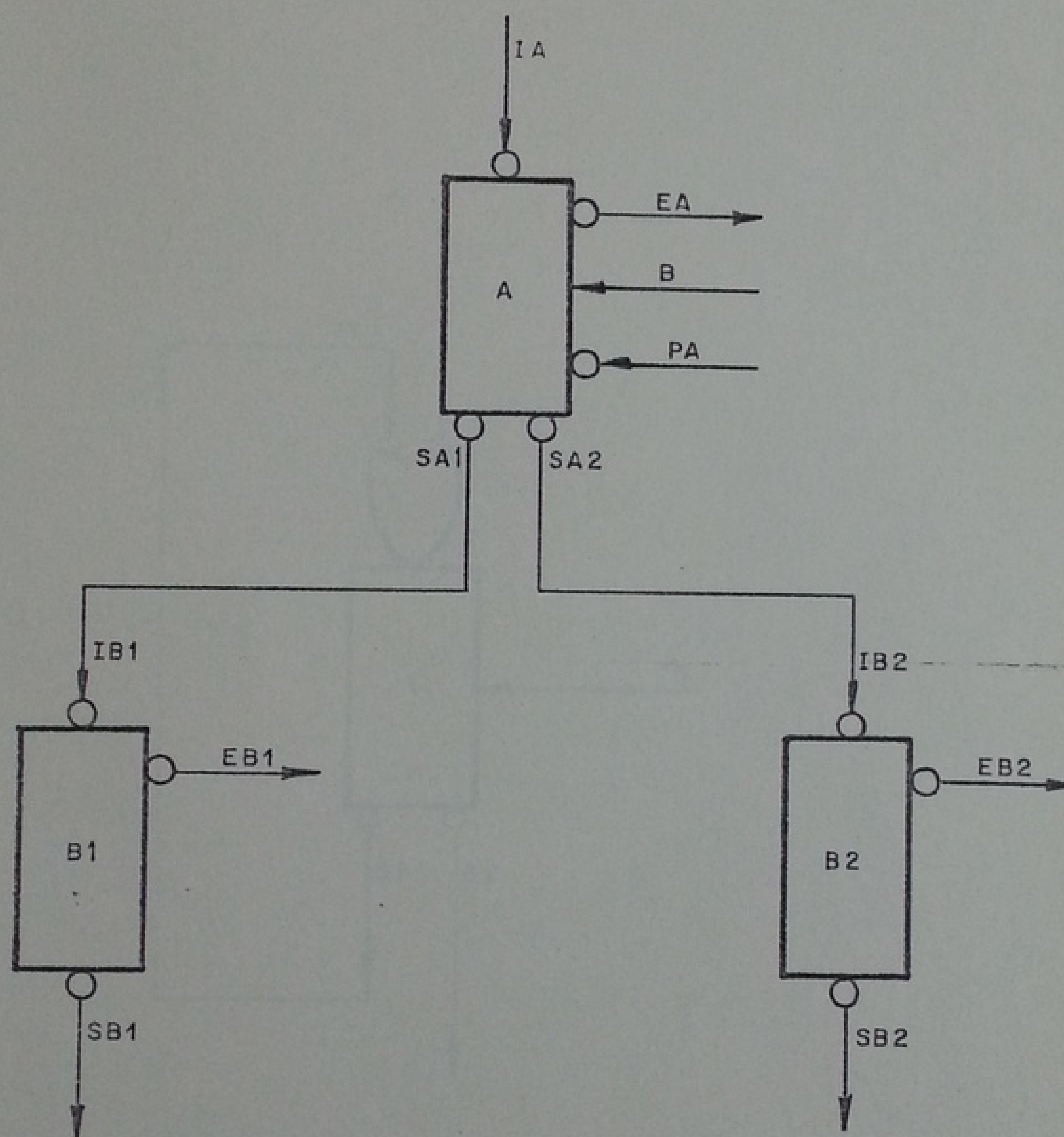
1. Sequencialização de eventos



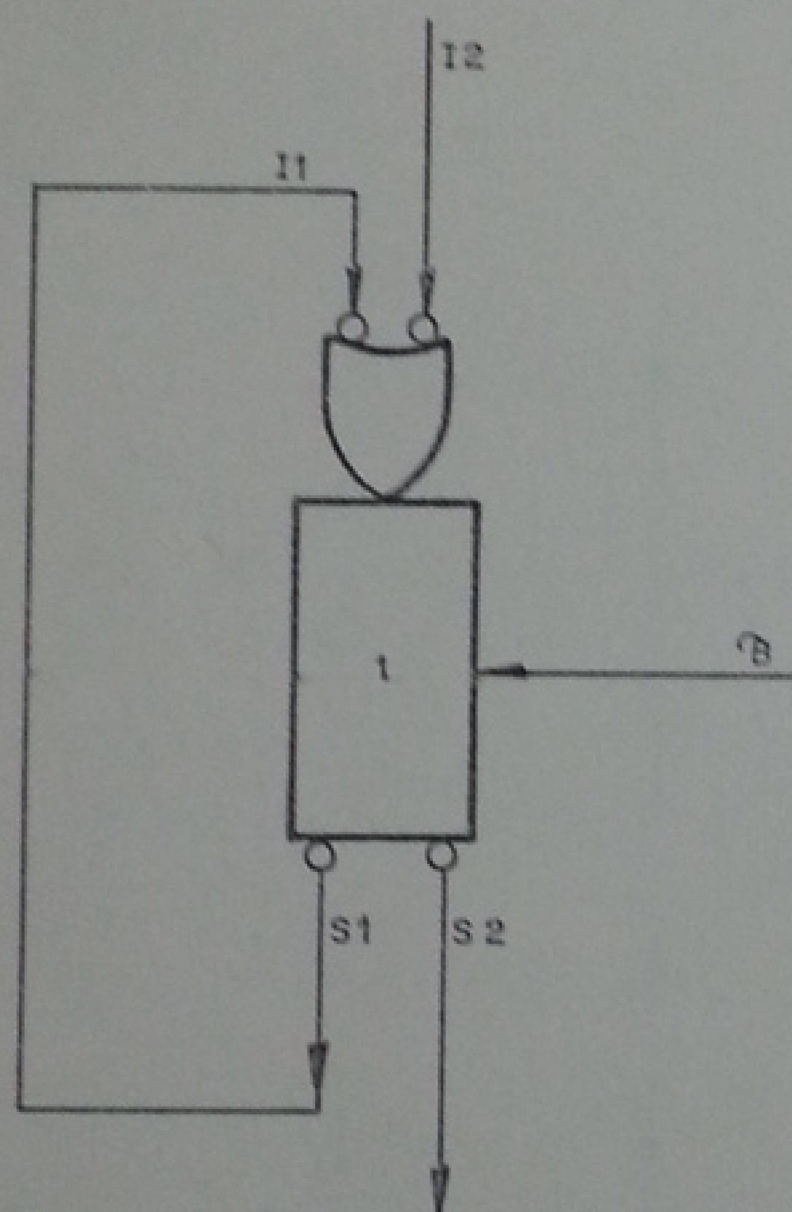
2. Ramificação de uma sequência incondicionalmente, em várias outras sequências simultâneas.



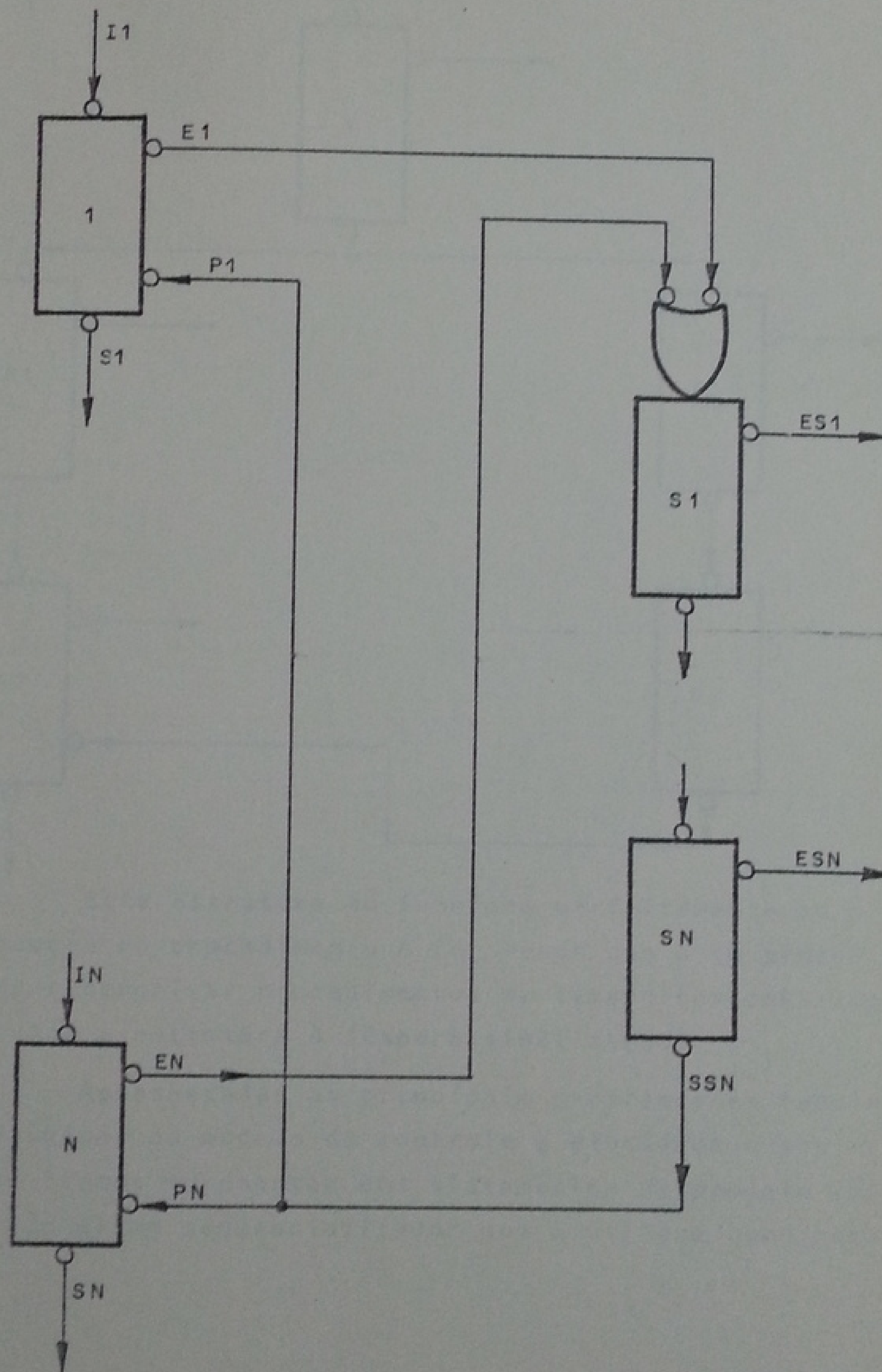
3. Ramificação condicional de uma sequência.



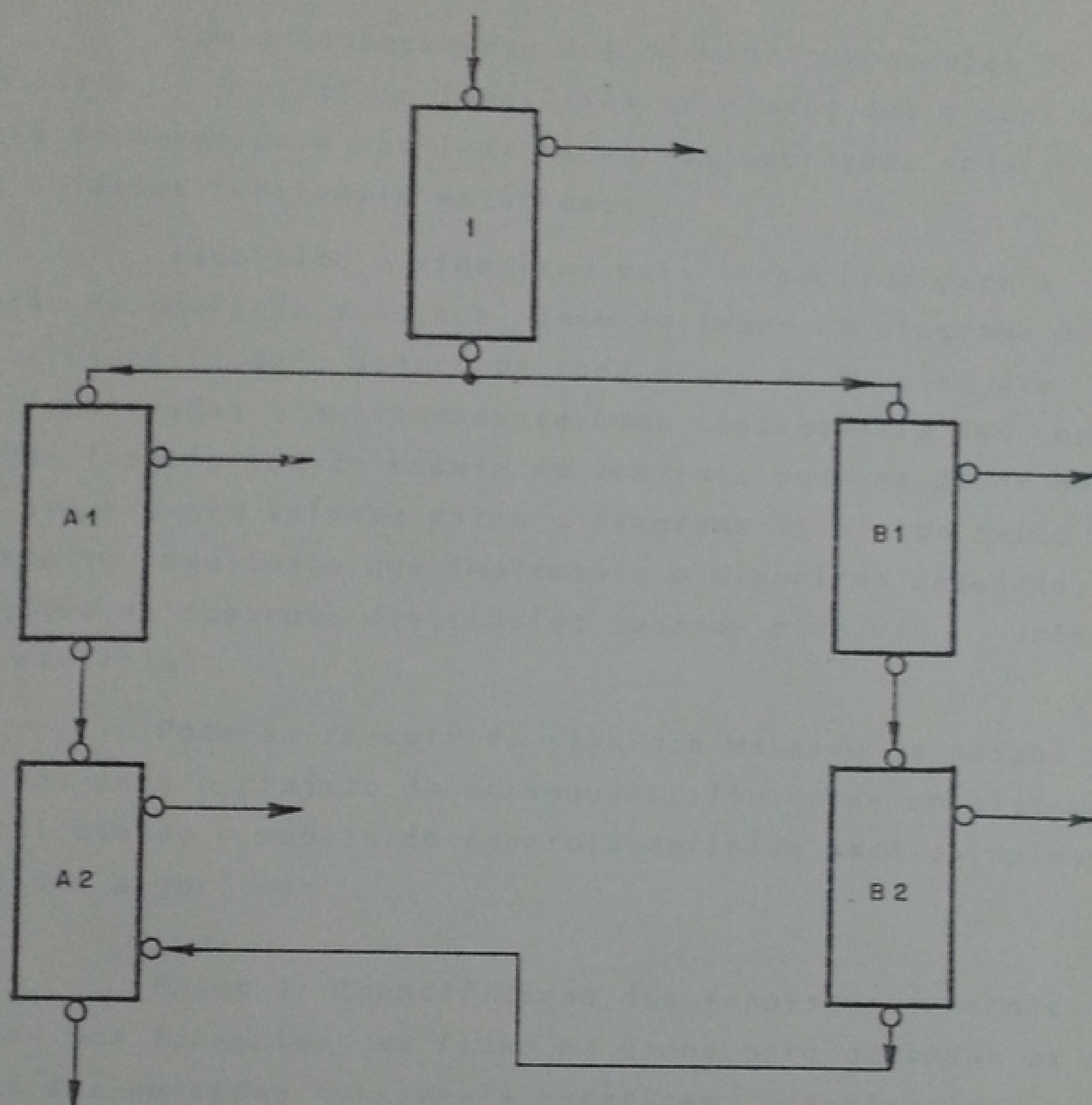
4. Espera sinalização (semáforo)



5. Execução de uma súbrotina.



6. Procedimentos concorrentes com tempos de execução diferentes.



Esta estrutura só funciona perfeitamente se o tempo de execução do procedimento A for menor que o do procedimento B. Para sincronizar procedimentos de tempos desconhecidos pode-se usar a estrutura 4 (Espera sinalização).

Apresentadas as principais propriedades funcionais e topológicas do módulo de controle e elucidado o seu funcionamento iremos apresentar uma sistemática de projeto lógico estruturado de um sequencializador que o utiliza como peça básica.

- O projeto

Com o conhecimento dos módulos componentes do fluxo de dados, o projetista identifica os sinais que o controle deverá fornecer para realizar as várias operações possíveis com as unidades funcionais estáticas.

Escolhido o algorítmo mais apropriado para a realização da operação desejada, deve-se fazer um diagrama de blocos, especificando dentro de cada bloco as funções que podem ser realizadas simultaneamente.Como consequência das propriedades topológicas do módulo de controle pode-se dizer que existe uma relação unívoca entre o diagrama de blocos feito e o circuito resultante que implementa o algorítmo desejado, num esquema de controle distribuído (usando o módulo universal de controle).

Pode-se resumir da seguinte maneira os passos necessários para o projeto de um sequencializadorde um sistema digital usando o módulo de controle definido para a implementação dos algorítmos.

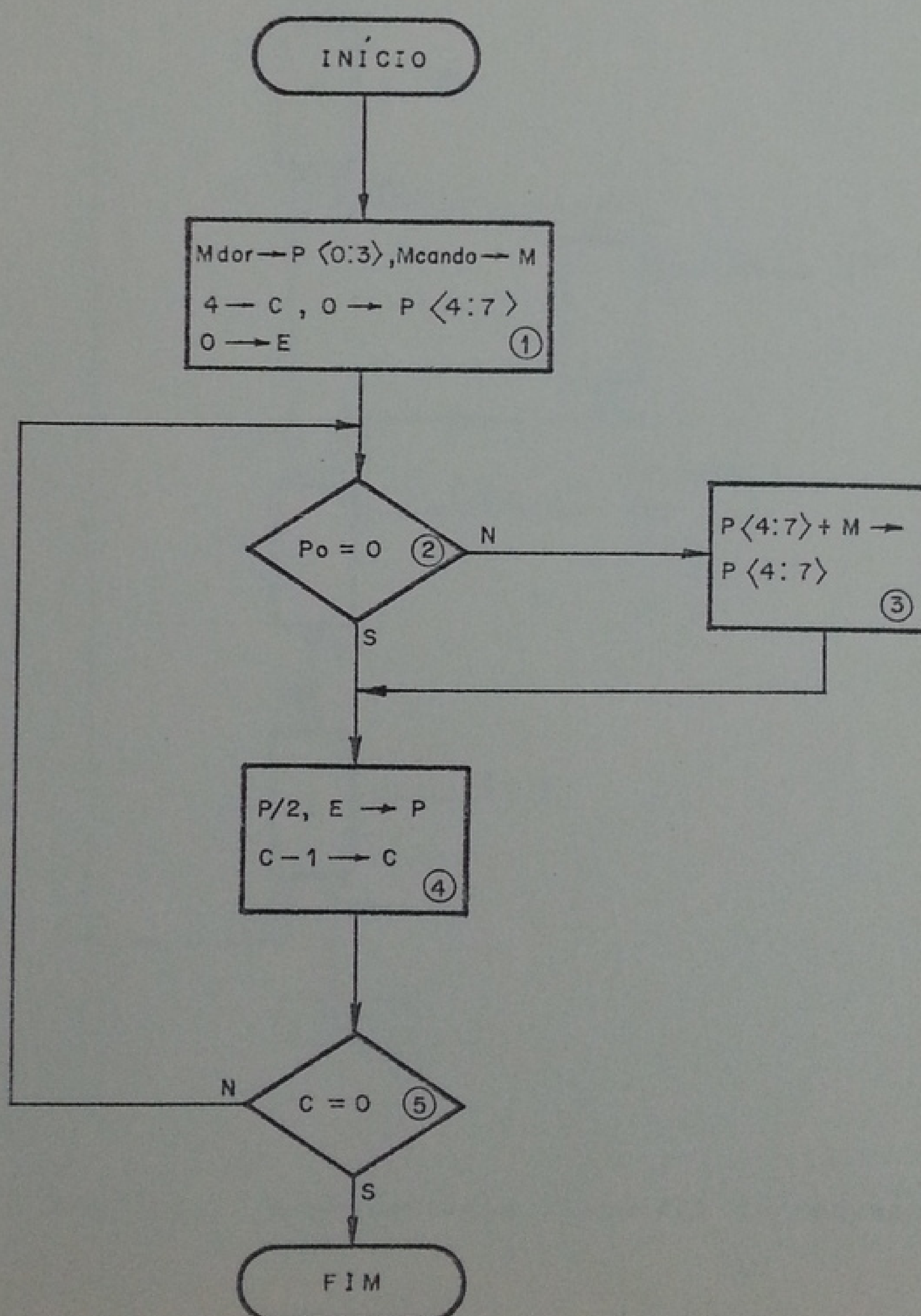
Passo I: Identificação dos sinais de controle que devem ser fornecidos ao fluxo de dados para comandar as operações das unidades funcionais estáticas. Conscientização do grau de paralelismoedo ciclo do fluxo de dados, para poder identificar claramente as operações que podem ser realizadas simultaneamente.

Passo II: Fazer um diagrama de blocos dos algorítmos a serem implementados pelo sistema, deixando bem claro qual a sequência de operações e quais as operações simultâneas em cada um dos estados. Aproveitando a relação unívoca existente entre o diagrama em blocos e o sequencializador, pode-se desenhar imediatamente o circuito final.

Exemplo 2: Projetar o sequencializador para o multiplicador do exemplo 1.

Os sinais de controle que o sequencializador deverá fornecer para o fluxo de dados já estão identificados na fig.II.1.Nesta figura também aparecem as variáveis de teste que serão utilizadas. A parte do identificador não será detalhada, por isso suporemos simplesmente que o sequencializador a ser projetado seja ativado por um sinal vindo do identificador.

O diagrama de blocos do algorítmo escolhido é o seguinte:



O circuito sequencializador resultante na sua representação simbólica é o seguinte:

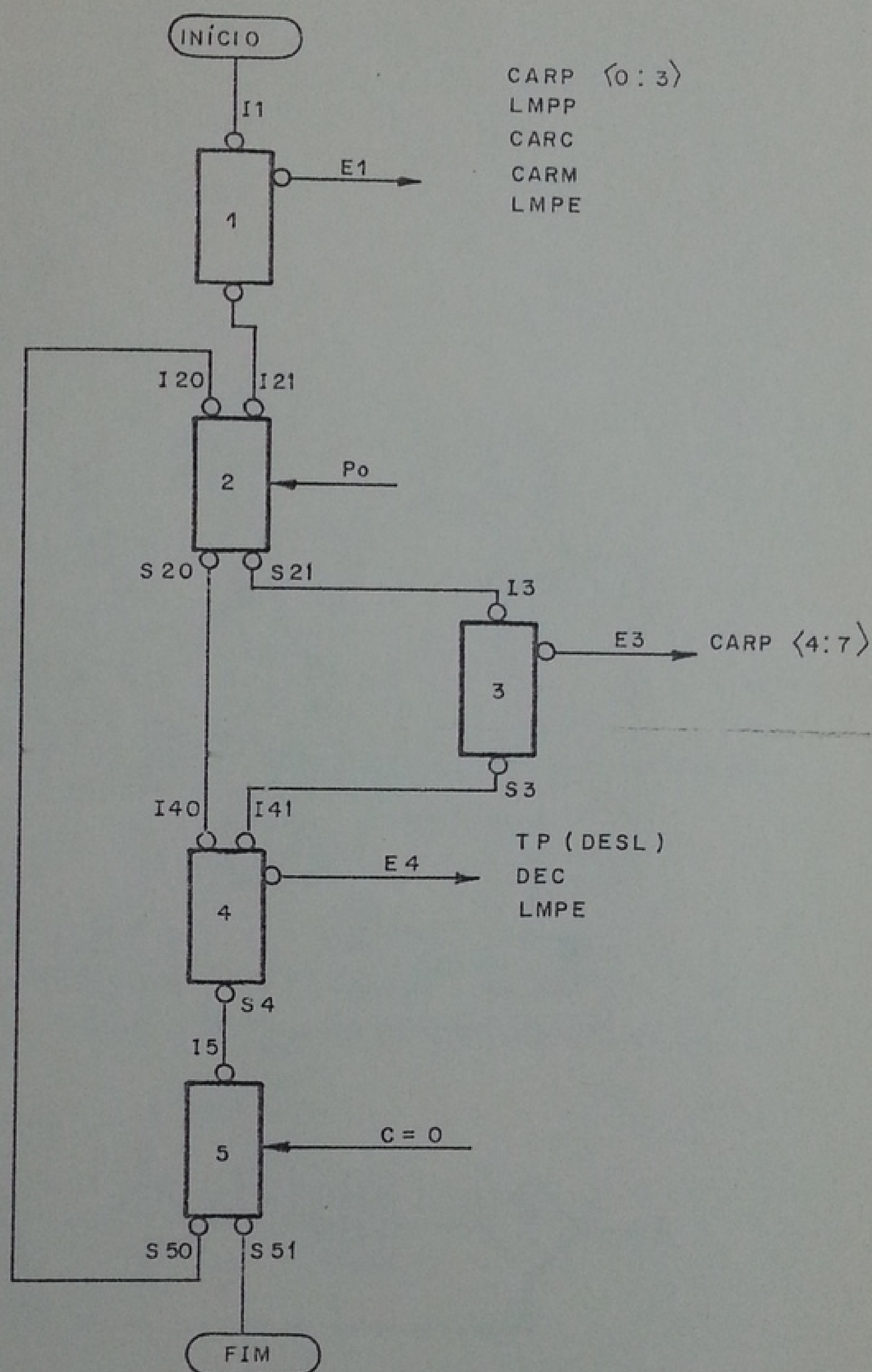


Fig. II.7. Representação simbólica do sequencializador



Fig. II.8. - Circuito completo do multiplicador

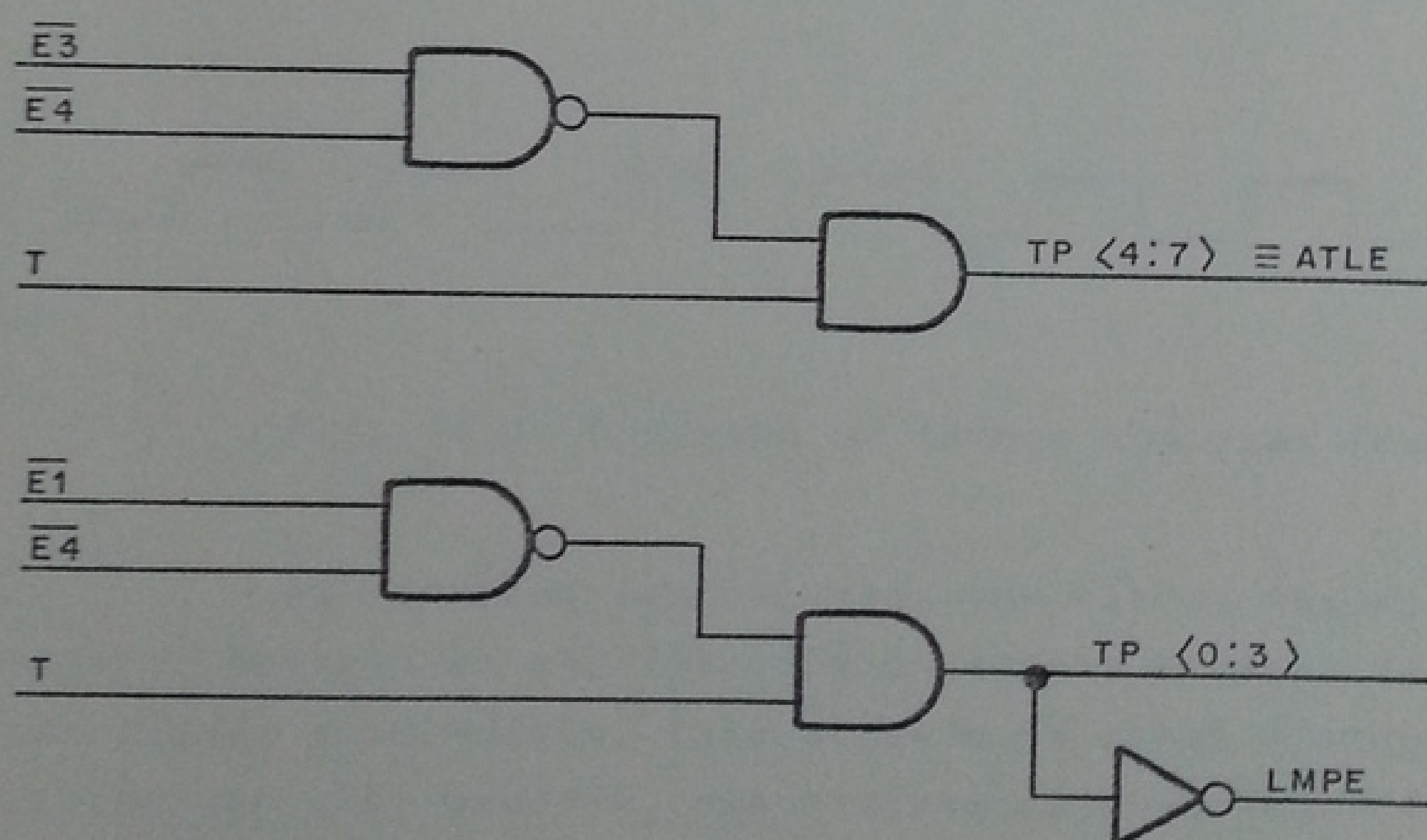
Portanto os sinais de comando do fluxo de dados são equacionados da seguinte forma:

CARP 0:3 = E1; TP = E4; CARP 4:7 = E3; LMPP = E1; CARC = E1;

DEC = E4; CARM = E1

Como pode-se observar estas equações expressam unicamente a forma de conectar as ordens geradas no controle e os sinais de comando do fluxo de dados.

Observação: (Fig.II.8) - não se levou em conta a polaridade dos sinais com o intuito de deixar o circuito na sua forma mais simples facilitando o seu entendimento. Também não foi desenhado a geração completa dos sinais TP (TP 4:7 ,TP 0:3) LMPE e ATLE, com o mesmo objetivo. Para que o circuito funcione corretamente esses sinais devem ser gerados da seguinte maneira:



onde T é o sinal base de tempo que coordena todas as mudanças de estado dos módulos de controle.

Observações:

O sinal de tempo T que irá comandar todas as transições de estado no sequencializador deve possuir um período compatível com o ciclo do fluxo de dados. Se existirem tempos de execução de operações muito diferentes, pode-se escolher um sinal de tempo de duas fases (ou mais), conseguindo-se assim uma velocidade maior para o sistema. Uma execução poderia durar um ciclo inteiro ou meio ciclo, dependendo da unidade funcional.

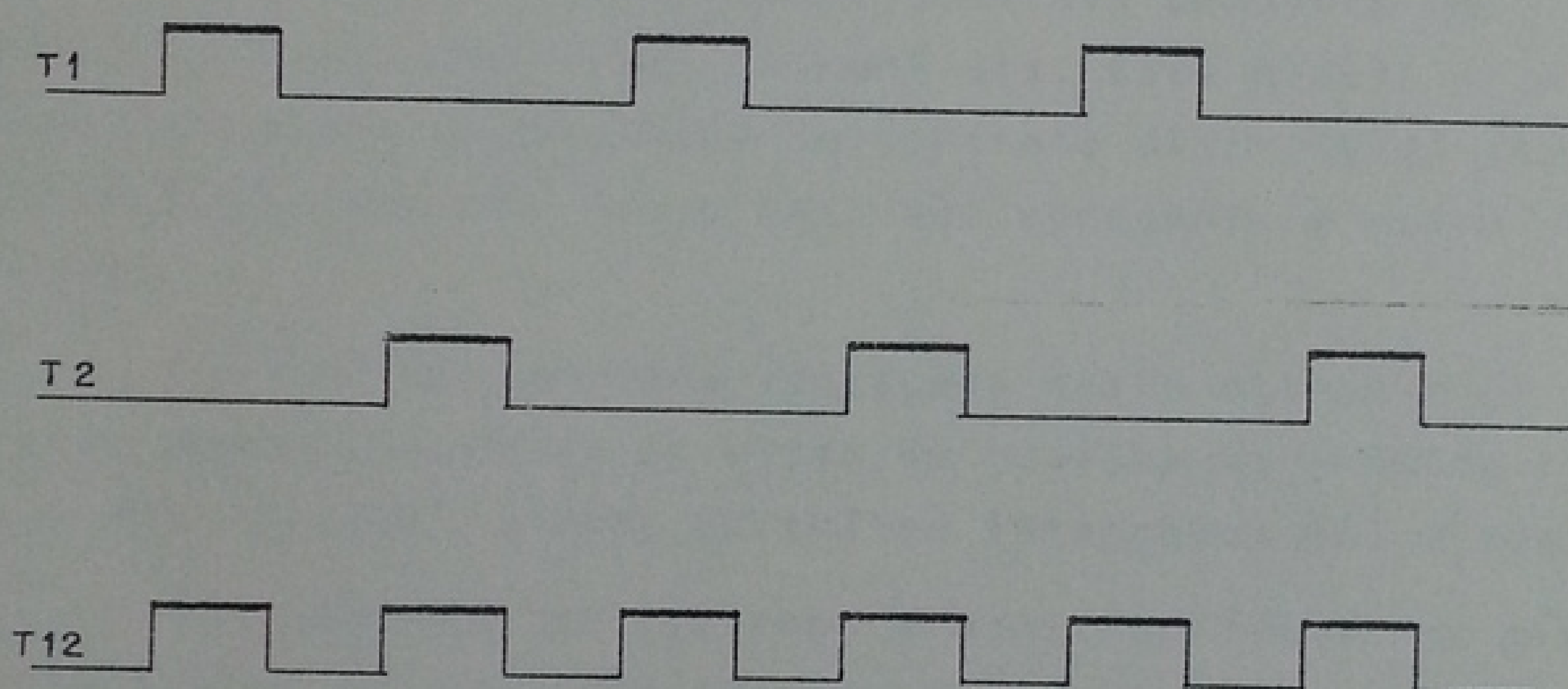


Fig. II.9. Diagrama de tempos com duas fases para T.

Num esquema implementado com várias fases, deve-se tomar o cuidado em escolher coerentemente o sinal de tempo a ser ligado a um módulo. Este depende do sinal ligado ao seu antecessor. Por exemplo, num sistema de duas fases (T1,T2) se tivermos tres blocos ligados em sequência, o primeiro determina operação com duração de um ciclo inteiro e é ativado por T1; o segundo é de duração curta, ativado com T1; o terceiro que também é de duração longa pode ser ativado pelo sinal T2.

A sistemática de projeto lógico estruturado de um sequencializador, que foi estabelecida, define clara e objetivamente o que vem a ser o controle de um sistema digital. Portanto qualquer outra sistemática de projeto lógico utilizada para o controle se resumirá numa implementação diferente das mesmas idéias básicas aqui apresentadas.

5.2. Projeto lógico do controle utilizando um núcleo de controle concentrado.

O procedimento de projeto que será apresentado a seguir se baseia num módulo de controle composto por vários circuitos integrados MSI disponíveis. Este módulo é de maior capacidade que o utilizado no controle distribuído (apresentado em 5.1), mas por outro lado não apresenta a mesma flexibilidade.

Uma das grandes vantagens desta sistemática de projeto, dentro do ponto de vista de realização prática atual, é que ela utiliza somente circuitos integrados MSI disponíveis.

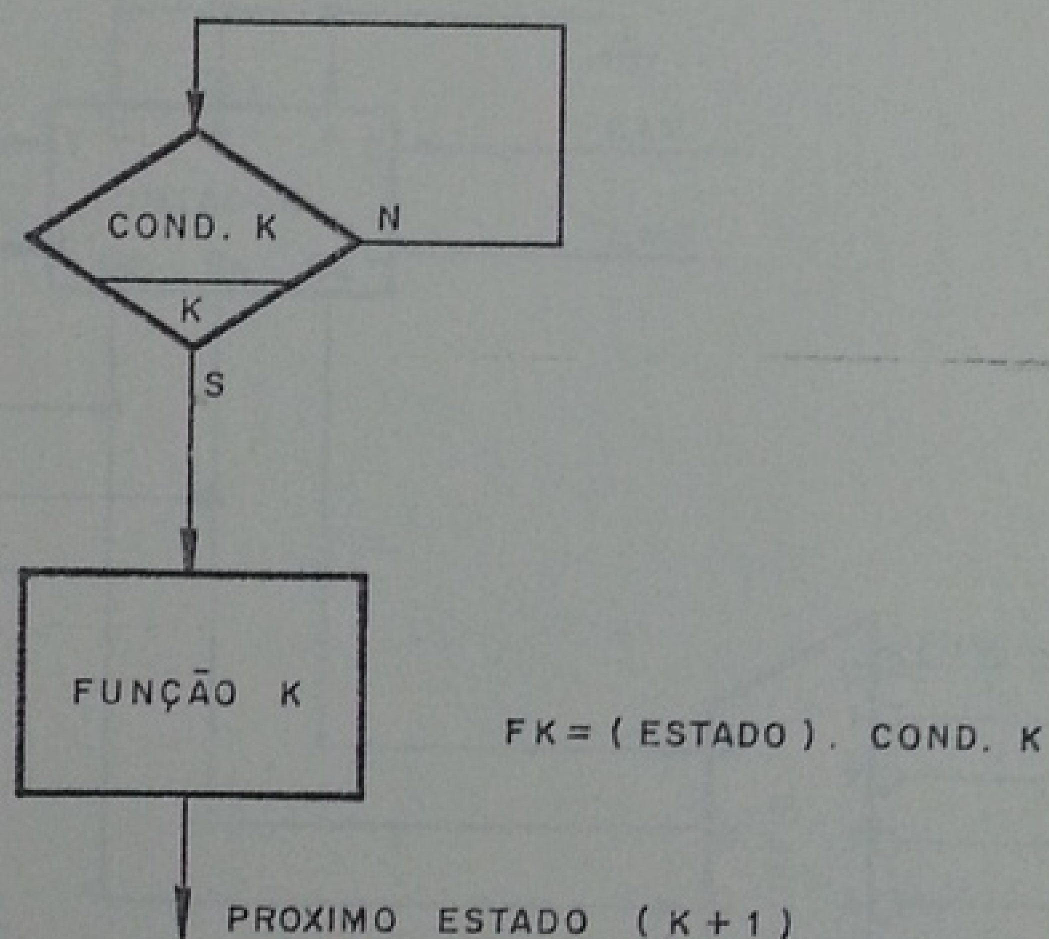
Características como grande facilidade de projeto. circuito de fácil entendimento, fácil e eficientemente documentável, manutenção simples e fácil depuração são as principais vantagens deste procedimento de projeto.

É claro que com tantos requisitos não se pretende conseguir um circuito de controle otimizado segundo o critério de menor número de portas lógicas, mas sim um circuito que utilize mais eficientemente, no projeto do controle, os módulos MSI existentes.

- O núcleo de controle

O núcleo básico utilizado é um circuito sequencial pulsado que apresenta 2^n estados, 2^n entradas e 2^n saídas. A cada estado corresponde uma dupla de sinais, um de entrada e outro de saída. Existe um sinal-padrão de tempo que comanda todas as transições entre os estados do circuito.

A cada estado do núcleo básico está associado um par Função (saída) - Condição (entrada) e uma regra fundamental: A condição necessária e suficiente para que se execute a Função é que a Condição seja verdadeira.



O núcleo de controle possui tres peças fundamentais: uma unidade contadora, uma unidade selecionadora (multiplexador) e uma unidade decodificadora. O contador é quem determina o estado em que o núcleo se encontra. O multiplexador é quem seleciona a condição que deve ser testada num determinado estado do circuito.

O decodificador é quem identifica a função a ser executada quando a condição for verdadeira.

Associada à execução de uma função existe uma transição de estado. Normalmente passa-se ao estado seguinte incrementando-se de uma unidade o contador. Esta sequência de estados pode ser alterada, permitindo assim desvios nas sequências de controle.

O contador de estados, o multiplexador de condições e o decodificador de funções são interligados da seguinte maneira para formarem o núcleo básico do controle (para esta representação adotou-se n=3).

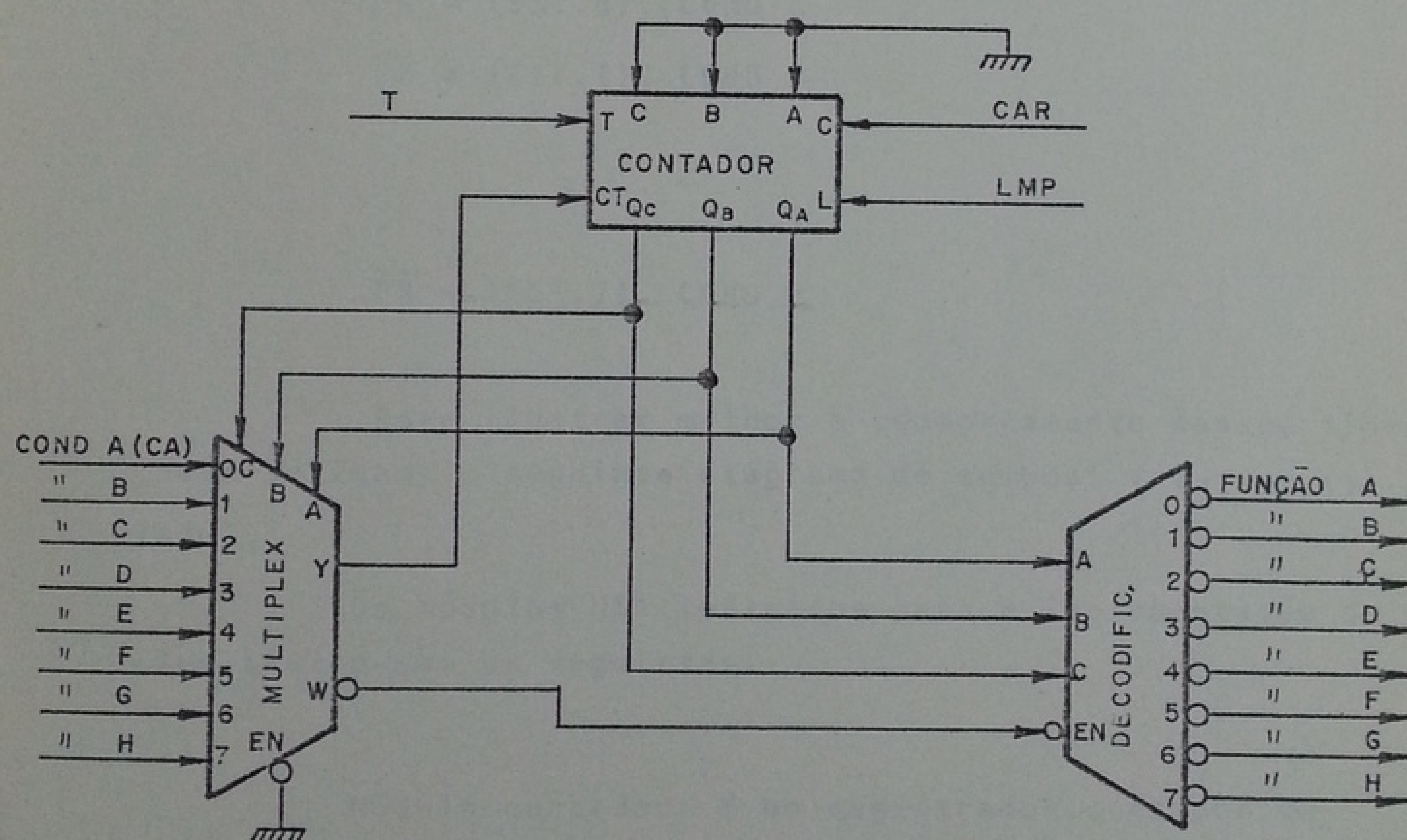


Fig. II.10. Núcleo básico de controle concentrado.

O núcleo de controle assim interligado é um sequencializador básico de operações. O estado inicial do sistema é $(000)_2$. Neste estado é determinada a execução da função A quando a condição A for verdadeira. Assim que a condição A se tornar verdadeira o contador muda para o estado $(001)_2$ em correspondência com o sinal T. Com o contador no novo estado, será determinada a execução da função B quando a condição B se tornar verdadeira e novamente ocorre uma mudança de estado do contador em correspondência com o sinal T. Dessa maneira todos os estados possíveis são percorridos.

As equações lógicas dos sinais que comandam as funções são as seguintes:

$$\overline{FA} = (EST.0).\ COND\ A$$

$$\overline{FB} = (EST.1).\ COND\ B$$

$$\vdots$$

$$\overline{FH} = (EST.7).\ COND\ H$$

Para ilustrar melhor o comportamento desses sinais, pode-se fazer o seguinte diagrama de tempos: (Fig. II.11 ao lado).

Os módulos MSI indicados para a implementação do núcleo básico são os seguintes:

Módulo contador: É um registrador contador de 4 bits que pode ser carregado com o conteúdo de 4 entradas paralelas em sincronismo com um sinal de tempo T ou o seu conteúdo ser incrementado de uma unidade, sempre em sincronismo com o sinal T.

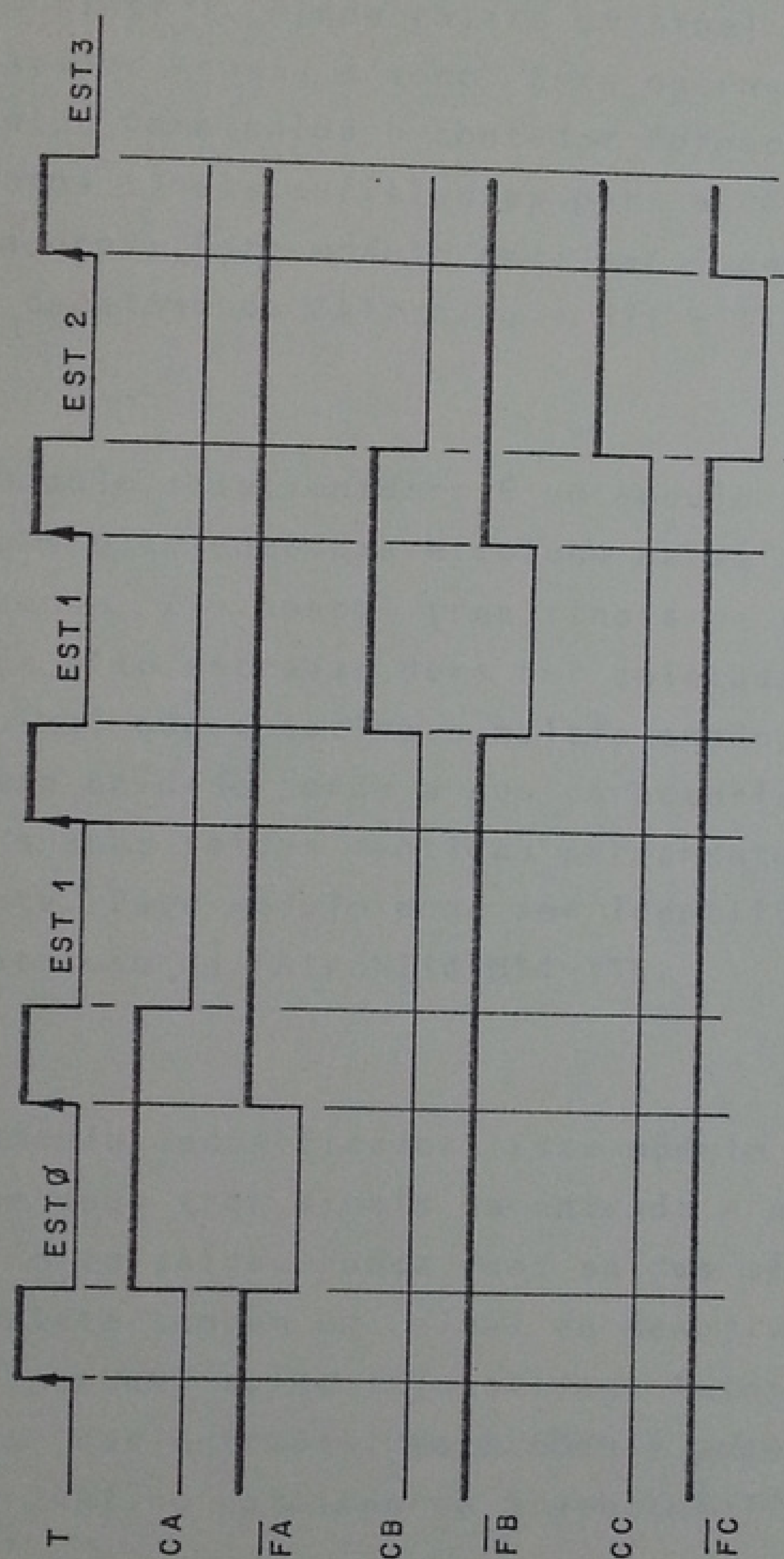


Fig.II.11. - Diagrama de tempo do núcleo de controle concentrado.

A distinção entre a carga ou a contagem é feita por um sinal de entrada do contador (CAR). Mesmo posicionado para contagem existe um sinal de entrada que pode bloquear o sinal de tempo T impedindo que o contador mude de estado (conte) à transição positiva do sinal T. Ainda existe um sinal que faz todos os bits do contador iguais a zero. Esta operação é realizada assíncronamente. Como saída o contador fornece os seus quatro bits, e outros sinais suficientes para a conexão de vários módulos contadores. Este módulo pode ser identificado pelo número 9316 no catálogo da FAirchild - MSI - TTL.

Módulo selecionador: É um módulo que seleciona uma entre as suas oito entradas e coloca na saída seu valor e o seu complemento. Ele possui tres sinais de seleção que indicam qual das oito entradas deve ser colocada na saída. Ainda existe um sinal que desativa o multiplexador. Quando este sinal atua esta unidade perde a sua característica selecionadora, sendo as suas saídas mantidas permanentemente com um valor constante. Este módulo pode ser identificado pelo número 74151 no catálogo da FAirchild-MSI-TTL.

Módulo decodificador: Este módulo identifica a configuração de seus tres sinais de entrada e emite um sinal em uma de suas oito saída. Nunca duas saídas são ativadas ao mesmo tempo. Existe também um sinal de desativação cuja atuação faz com que nenhuma saída seja ativada independentemente da configuração das entradas. Este módulo pode ser identificado pelo número 7442 no catálogo da Signetics-TTL.

Tendo sido apresentado um esquema simplificado do núcleo básico de controle concentrado e explicado o seu princípio de funcionamento. pode-se passar à apresentação das suas propriedades funcionais.

- Propriedades do núcleo de controle

Serão apresentadas apenas as propriedades mais fundamentais e úteis para a elaboração de algorítmos através do núcleo básico. Não se pretende esgotar neste relato todas as possibilidades do circuito apresentado.

1. Execução incondiconal de uma função: Uma determinada função pode ser executada incondicionalmente. Para isso basta fixar a entrada do multiplexador de condições associada a essa função em nível lógico "1" (verdade).

2. Desvio da sequência normal: Pode-se passar de um determinado estado para outro, sem realizar as funções associadas aos estados intermediários. Para isso basta carregar o contador de estados com o valor do estado que se deseja atingir. Esta operação é possível através da ação de uma função (pulo) na entrada da unidade contadora que a posiciona para contagem (CAR). A mudança de estado no contador ainda continua sendo em sincronismo como o sinal T. O valor do estado a ser carregado deve ser gerado por um circuito a parte que possui como entradas sinais de funções (pulos), sinais indicadores de estado e sinais de identificação de comandos externos ao núcleo. As saídas deste circuito devem ser ligadas às entradas para carga paralela (a,b,c,) do contador de estados.

Convém ressaltar que o circuito identificador do controle se comunica com o sequencializador implementado por este núcleo, através deste circuito de geração de estados. O identificador decodifica o comando externo emitido e fornece sinais para o gerador de estados. Este circuito por sua vez, aguarda a execução de uma determinada função para carregar no contador o estado inicial da sequência que executa o comando requisitado.

Depois de apresentar a possibilidade de desvio da sequência normal de estados, pode-se dizer genericamente que a equação de mudança de estado do circuito é a seguinte:

$$E_{i+1} = \begin{cases} E_i + 1 & \text{se } FP = 0 \\ P & \text{se } FP = 1 \end{cases}$$

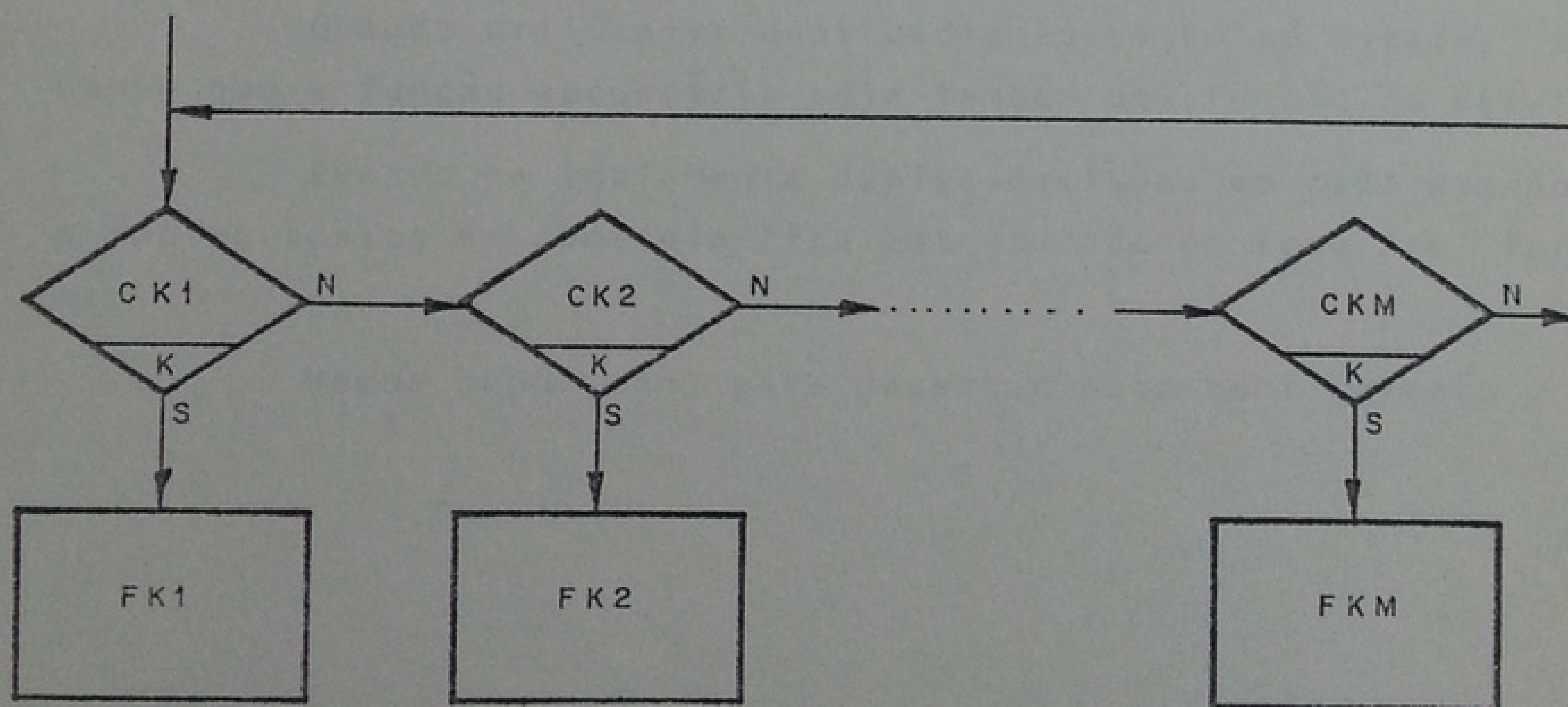
onde E_i conteúdo do contador no instante i (número binário)

E_{i+1} conteúdo do contador no instante i+1

P valor do estado a ser atingido

FP função cuja execução determina um desvio na sequência normal de estados.

3. Multiplas decisões: Pode-se associar a um estado do circuito mais de uma função e mais de uma condição. Neste caso a regra fundamental utilizada para a determinação da execução de uma função é expressa pelas seguintes equações:



$FK1 = (ESTK) . CK1$

$FK2 = (ESTK) . \overline{CK1} . CK2$

$FKM = (ESTK) . \overline{CK1} . \overline{CK2} \ldots . CKM$

Para que o núcleo básico do controle seja capaz de analisar várias condições num mesmo estado é necessário dividir o multiplexador de condições e o decodificador de funções em duas partes. Para exemplificar, pode-se usar um esquema que leva em conta duas condições em cada estado do sistema. A primeira condição de cada estado será conectada à parte do multiplexador de condições que comanda a primeira parte do decodificador de funções. Se esta condição for verdadeira será executada a função que estiver endereçada através da primeira parte do decodificador. Se a condição for falsa será ativada a segunda parte do multiplexador de condições que interpretará a segunda condição associada a esse estado. Se esta nova condição for verdadeira será executada a função endereçada através da segunda parte do decodificador de função. Se uma das duas funções for executada, é provocada uma mudança de estado, se não, o controle permanece neste estado a espera de que uma das duas condições se torne verdadeira.

A primeira parte do multiplexador de condições trata das condições primárias e a segunda parte das condições secundárias. A primeira parte do decodificador de funções emite as funções primárias e a segunda parte as funções secundárias.

Quando utiliza-se duas decisões em um só estado, é comum que a função secundária seja também uma função de desvio.

Quando se implementa duplas decisões em cada estado, o núcleo básico de controle fica estruturado da seguinte forma:

Vamos supor n=3 para desenhar esta configuração.

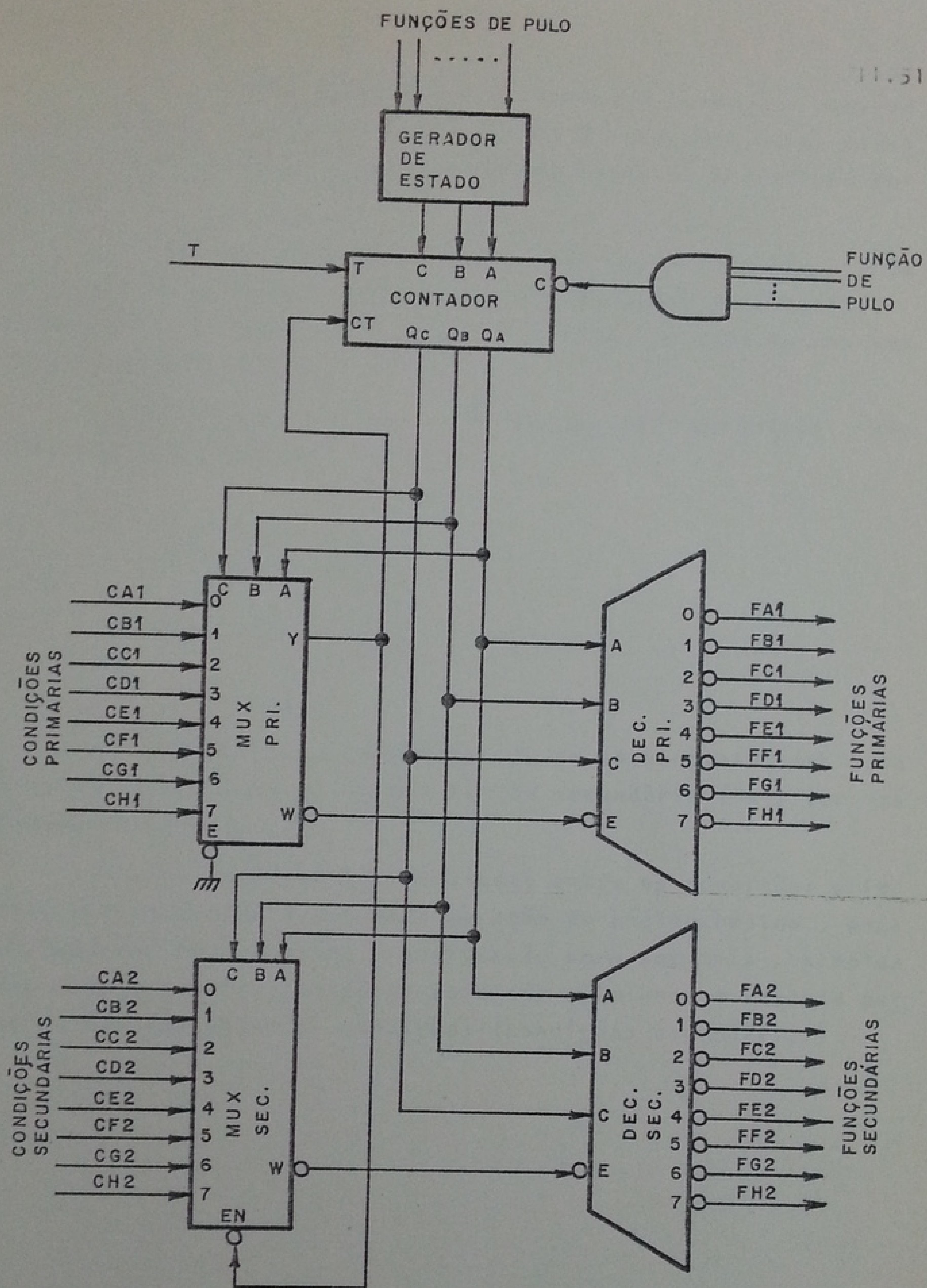


Fig.II.12. - Núcleo básico com múltiplas decisões

4. Ramificação; A uma determinada condição pode-se associar duas diferentes funções. Uma é realizada se a condição for verdadeira e a outra será realizada se a condição for falsa.

Esta propriedade é uma consequência imediata da propriedade anterior. Basta fazer as condições secundárias iguais ao complemento das condições primárias para termos uma ramificação de sequência.

Desta forma a regra de decisão para um estado que possua ramificação é a seguinte:

$$\overline{FK1} = (EST\ K)\ CK$$

$$\overline{FK2} = (EST\ K)\ \overline{CK}$$

Novamente, para estados de um sequencializador que possuam ramificação é comum a função secundária (FK2) ser uma função de desvio.

Devido à relação existente entre as condições primárias e as secundárias, a configuração do núcleo básico para implementar este tipo de ramificação pode ser mais simples que o de multiplas decisões. Pode ser eliminada a segunda parte do multiplexador de condições (condições secundárias).

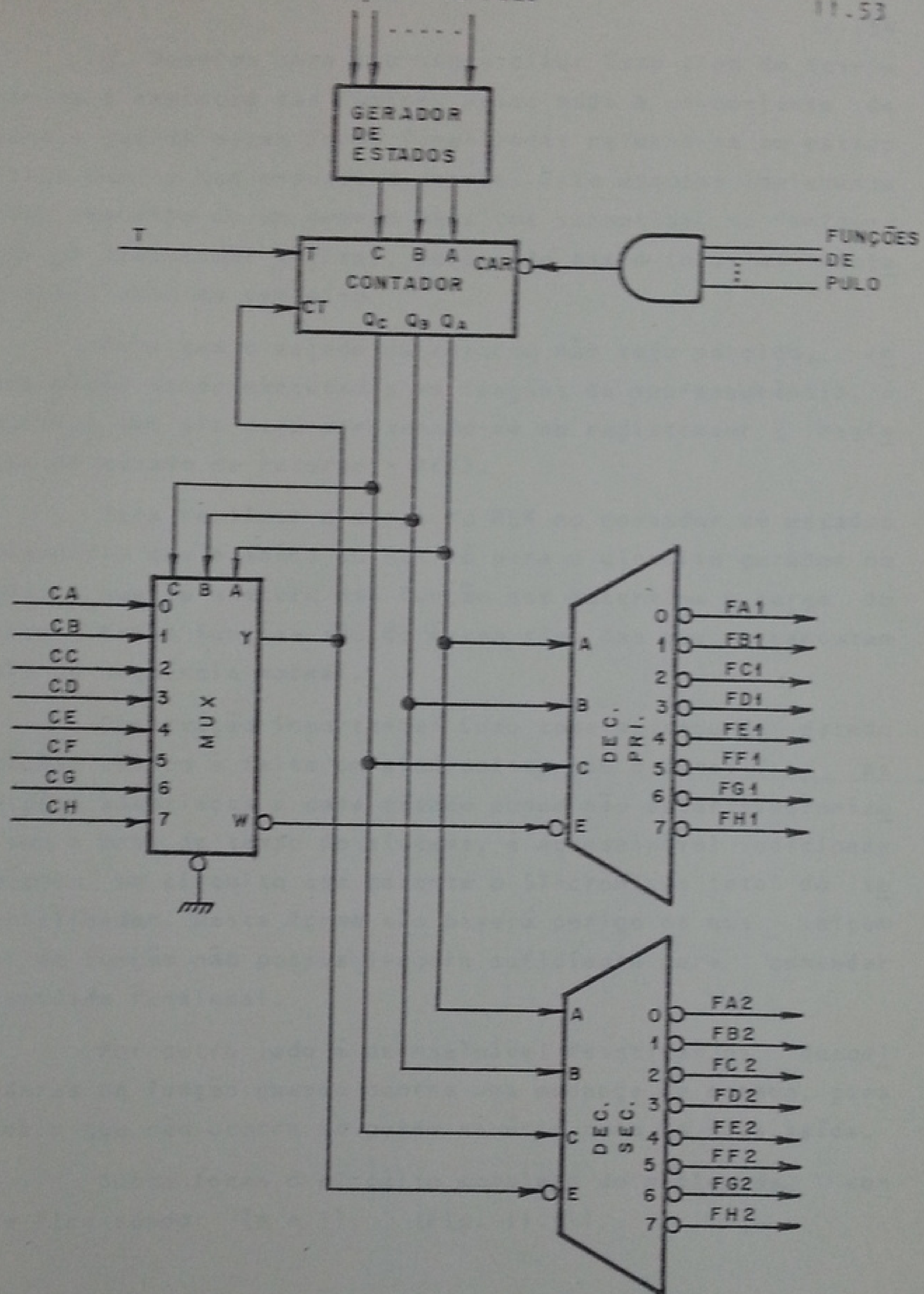


Fig.II.13. - Núcleo básico com ramificação

5. Desvios para sub-sequências: Este tipo de desvio determina a execução das funções associadas a um conjunto de estados e quando estas forem finalizadas retorna-se ao estado seguinte aquele que ordenou o desvio. Este esquema implementa o mesmo conceito de um desvio para uma subrotina, no "software" de um computador digital, possuindo assim todas as vantagens associadas ao conceito.

Para que o estado de retorno não seja perdido, enquanto estão sendo executadas as funções da sub-sequeência, é necessário que ele seja armazenado em um registrador (registrador de estado de retorno - RER).

Para realizar a carga do RER no contador de estados é necessário que a saída do RER vá para o circuito gerador de estados e que se execute uma função que determine a carga do contador. Estas funções são do mesmo tipo das que executam desvio de sequência normal.

Observação importante: Como toda mudança de estado do núcleo básico é feita em sincronismo com o sinal T e as condições associadas a cada estado podem não estar sincronizadas com a base de tempo do sistema, é aconselhável adicionar ao núcleo um circuito que garanta o Sircronismo total do sequencializador. Desta forma não haverá perigo de que algum sinal de função não possua largura suficiente para comandar uma unidade funcional.

Por outro lado é aconselhável desativar os decodificadores de função quando ocorre uma mudança de estado, para garantir que não ocorra um pulso espúrio numa de suas saída.

Desta forma o circuito completo do núcleo de controle fica sendo: (n = 3). (Fig. II.14).

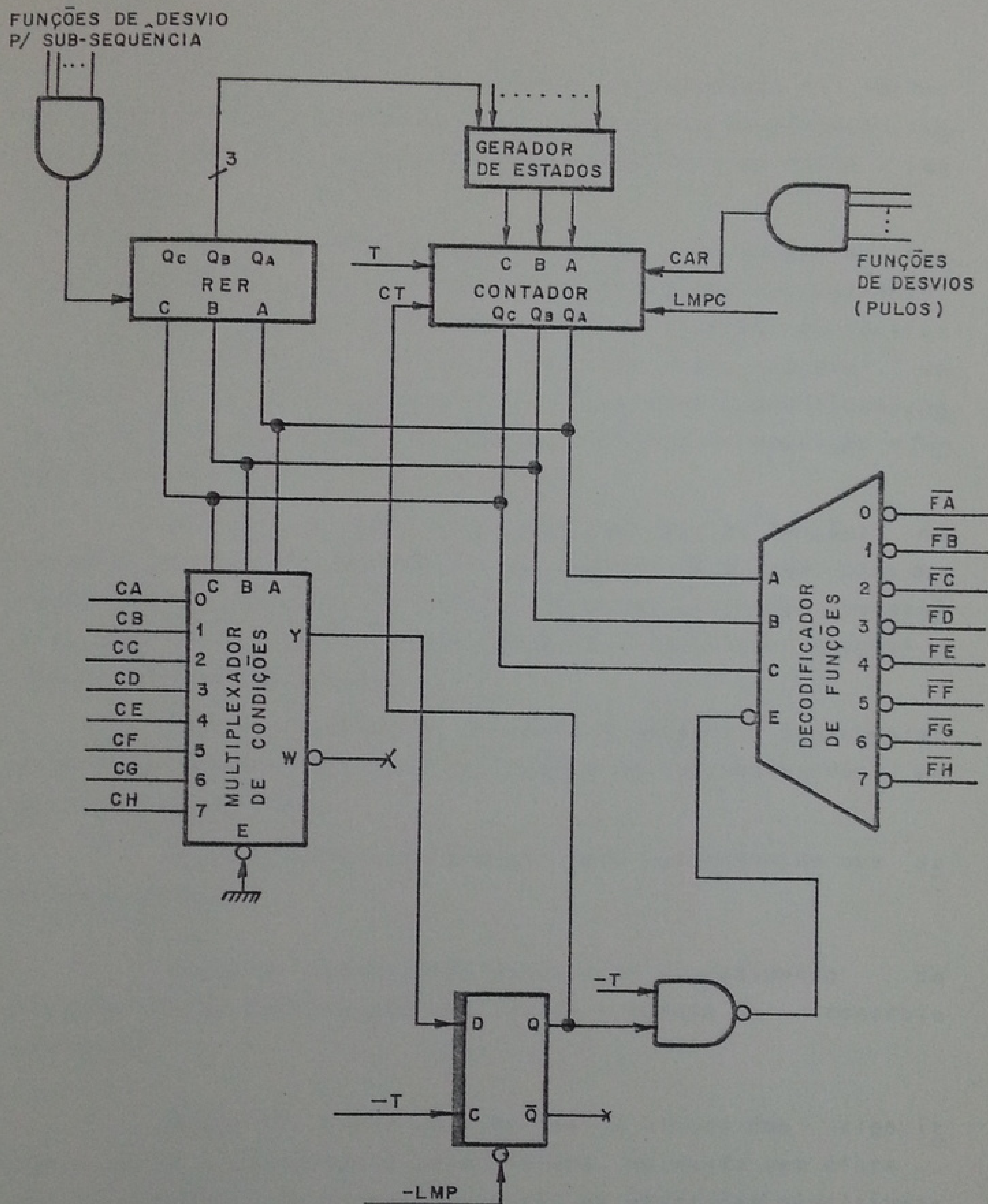


Fig.II.14. - Núcleo básico com circuito de sincronismo e RER.

- O projeto

Identificadas as principais propriedades do núcleo de controle pode-se descrever uma sistemática de projeto estruturado para um sequencializador, levando-se em conta a sua utilização.

Devido às propriedades funcionais do núcleo de controle concentrado, pode-se perceber que o mesmo procedimento usado para o projeto utilizando o módulo universal de controle pode ser novamente aplicado. Para isto basta escrever um diagrama de blocos dos algorítmos, ligeiramente modificado, para se poder identificar fácilmente os sinais de condição e função do núcleo básico.

O algorítmo escolhido para executar as funções do sistema deve ser representado num diagrama de blocos, onde apareçam identificados nos diversos blocos as condições necessárias para a realização de uma função e as operações envolvidas na execução da função.

Com este núcleo de controle é possível implementar grande parte dos algorítmos que podem ser representados em um diagrama de blocos.

O procedimento de projeto pode ser resumido nos seguintes passos:

Passo I: Idêntico ao passo I do procedimento de projeto lógico estruturado utilizando o módulo de controle universal.

Passo II: Fazer um diagrama de blocos dos algorítmos a serem implementados pelo sistema, deixando bem clara qual a sequência de operações, quais as operações simultâneas em cada um dos estados, as condições e funções associadas a cada estado; identificar claramente os desvios da sequência normal e os desvios para sub-sequências.

Com o diagrama de blocos feito, determinar o número de estados necessários para a sua implementação. Desenhar o núcleo básico de controle contendo somente as peças utilizadas para a implementação dos algoritmos. Do diagrama de blocos identificar as duplas condições-função para cada estado, desenhando quais são as variáveis de teste ou indicadores de estado que devem ser ligados na entrada do multiplexador de condições e quais as funções que deverão atuar sobre cada um dos sinais de comando do fluxo de dados.

A melhor maneira de tornar claro um procedimento é fazer um exemplo.

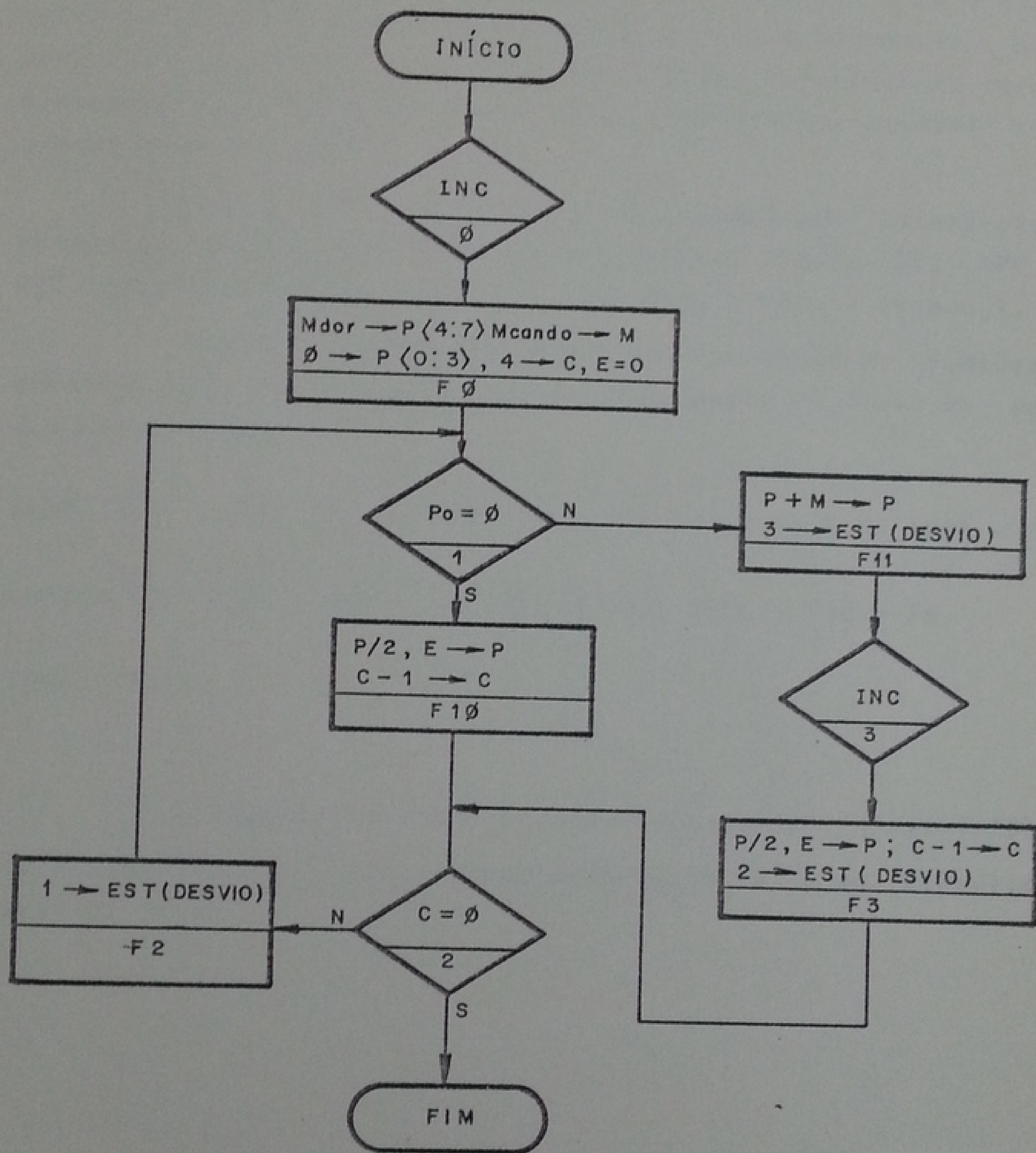
Exemplo 3: Projeto do sequencializador do multiplicador binário do exemplo 1. (Usar o mesmo fluxo de dados).

O diagrama de blocos do algoritmo, já devidamente adaptado, fica sendo : (figura ao lado).

Do diagrama de blocos feito, descobre-se que 3 estados são necessários para a implementação do algoritmo, portanto um núcleo básico de controle com n = 2 é suficiente. Existem três funções que determinam um desvio da sequência normal. Estas funções com o valor dos estados a serem atingidos pela execução de cada uma delas determinam as entradas e saídas do gerador de estado. Este é um circuito lógico combinatório facilmente sintetizável pelos métodos tradicionais.

Dependendo do algoritmo, o circuito gerador de estados pode se tornar complexo e grande. Para estes casos é aconselhável sintetizá-los através de multiplexadores ou mesmo através de memórias de conteúdo fixo (ROM).

As variáveis de teste utilizadas (estas explicações referem-se a Fig. II.15) são P_0 e C = 0. O estado Ø é executado incondicionalmente e a função FØ comanda os seguintes sinais CARP 0 : 3; CARM, LMPP, CARC. A condição associada ao estado 1 é P_0 . Neste estado existe uma ramificação. Se P_0 for igual a zero é executada a função F11. Se P_0 for igual a um é executada a função F10. A função F11 é responsável pelos seguintes sinais :


INÍCIO
INC
Ø
Mdor → P ⟨4:7⟩ Mcando → M
Ø → P ⟨0:3⟩ , 4 → C, E=0
F Ø
Po = Ø
1
N
S
P + M → P
3 → EST (DESVIO)
F11
P/2 , E → P
C − 1 → C
F 1Ø
INC
3
P/2, E → P ; C − 1 → C
2 → EST (DESVIO)
F 3
1 → EST(DESVIO)
F 2
N
C = Ø
2
S
FIM

CARP 4:7 ; ATLE. Esta função também determina um desvio para o estado 3. A função F1Ø é responsável pelos seguintes sinais: DESL, LMPE, DEC.

A condição associada ao estado 2 é C = Ø . Se o contador do fluxo de dados (C) for igual a zero a sequência de controle é encerrada. Se o contador (C) for diferente de zero é executada a função F2. Esta função simplesmente executa um desvio para o estado 1.

A função do estado 3 é F3 e é executada incondicionalmente. Ela é responsável pelos seguintes sinais: TP, LMPE, DEC. Esta função também determina um desvio para o estado 2.

Desta forma, as equações que relacionam as funções emitidas pelo controle e os sinais de comando do fluxo de dados são as seguintes:

CARP 0:3 = FØ ; TP = F1Ø + F3 ; CARP 4:7 = F11; LMPP = FØ

CARM = FØ ; DEC = F1Ø + F3; ATLE = F11; LMPE = F1Ø + F3,

CARC = FØ.

O sequencializador obtido é o seguinte: (Fig.II.15)
Para este circuito vale a mesma observação feita ao circuito da Fig. II.8.

Fig. II.15. - Circuito multiplicador com controle concentrado, sem a parte de sincronismo.

5.3. Projeto lógico de controle utilizando um núcleo de controle microprogramado

Para poder ser apresentada a estrutura de um núcleo de controle microprogramado, é necessário que se defina claramente o que vem a ser microprogramação.

De uma maneira ampla, pode-se definir microprogramação[5] como sendo uma técnica para projetar e implementar o controle de um sistema digital, como uma sequência de sinais de controle que interpreta funções de processamento fixas ou dinamicamente variáveis. Estes sinais de controle,organizados em palavras e armazenados numa memória de controle fixa ou dinamicamente variável, determinam o estado dos sinais que controlam o fluxo de informações entre as unidades funcionais e as mudanças de estado do sistema.

Relativamente aos objetivos do projeto de um sistema digital pode-se esclarecer ainda mais este conceito. Microprogramação é um método para controlar sistemas digitais que se baseia no estado das linhas de controle do sistema armazenados numa memória organizada em palavras. Estas palavras contêm as informações básicas para o controle do sistema. Cada palavra especifica as operações que as unidades funcionais do fluxo de dados deverão realizar em um ou mais ciclo de máquina. Portanto, lendo estas palavras numa sequência predeterminada e decodificando-as, pode-se ativar um subconjunto das linhas de controle que irão ativar um subconjunto de unidades em todo o sistema. A sequência de leitura das palavras de controle pode ser vinculada pela própria palavra que está em execução ou obedecer a uma regra pré-fixada.

Nesta estrutura de controle pode-se identificar também as partes já definidas de uma unidade de controle de um sistema digital. O decodificador determina qual o conjunto de linhas de controle que deverão ser ativadas.

O sequencializador (sincronizado por pulsos de tempo) é projeto para abrir e fechar uma série de portas lógicas espalhadas pelo sistema para permitir que as informações fluam de uma unidade funcional para outra.

A lógica de decisão executa certos testes sobre algumas variáveis de estado e o resultado destes testes podem alterar a seleção da próxima palavra de controle que irá controlar o sistema no próximo ciclo.

A esta altura, convém definir mais claramente, o que vem a ser ciclo de máquina num núcleo de controle microprogramado. O ciclo de controle é o intervalo de tempo compreendido entre a busca de uma palavra na memória de controle e o término de sua execução, quando não existe sobreposição entre a busca e a execução de microinstruções. Mais precisamente, o ciclo de controle determina a frequência de execução das palavras de controle. Dentro de um ciclo de controle pode-se ter vários sinais de tempo que sincronizam todas as operações executadas.

Pode-se definir também como sendo uma microinstrução, o conjunto de operações executadas dentro de um ciclo completo de controle. Uma microinstrução pode ser composta por uma ou mais palavras de controle.

Considerando-se as características de "hardware" de um sistema digital microprogramado convém examinar o projeto e a implementação das microinstruções antes de exemplificar um núcleo de controle microporgramado. O projeto de uma microinstrução consiste no estabelecimento das informações necessárias para controlar os recursos (unidades funcionais) do sistema e o agrupamento e divisão dessas informações dentro da palavra de controle. A implementação estabelece a maneira pela qual o sistema executa uma microinstrução.

- Projeto das microinstruções

É de fundamental interesse no projeto das microinstruções o número de recursos que cada uma delas controla. Em relação a este aspecto, as microinstruções são usualmente classificadas como vertical ou horizontal. Uma microinstrução vertical efetua operações simples do tipo: carrega, soma, armazena, pula, etc... Elas frequentemente recodificam instruções de uma linguagem de máquina, contendo um código de operação e dois ou mais operandos. O tamanho de uma microinstrução vertical varia tipicamente de 12 a 24 bits.

Uma microinstrução horizontal, em contraste, controla muitos recursos do sistema que operam simultaneamente (em paralelo). Uma simples microinstrução horizontal é capaz de controlar as operações simultâneas e independentes de uma ou mais unidades lógicas e aritméticas, leitura e escrita de memória, geração condicional do próximo endereço, etc... Enquanto que a microprogramação horizontal tem a grande vantagem de utilizar eficientemente o "hardware", desenvolver microprogramas horizontais que utilizem de maneira ótima os recursos de "hardware" é uma tarefa difícil. Devido ao fato de uma microinstrução horizontal controlar múltiplos recursos, ela contém mais informações que a vertical e, portanto, possui um tamanho maior; microinstruções horizontais com 64 ou mais bits são comuns. A característica determinante entre microinstruções verticais e horizontais é o número de recursos controlados simultaneamente.

O grau de codificação numa palavra de controle afeta o seu tamanho, sendo assim frequentemente confundido com a característica vertical-horizontal. No projeto mais simples de uma microinstrução não existe nenhuma codificação nos bits da palavra. Cada bit controla uma unidade funcional ou uma operação no sistema.

Num projeto com codificação direta (um nível de codificação) os bits que controlam recursos ou operações mutuamente exclusivas, tais como operações que uma unidade lógica e aritmética pode fazer ou o endereçamento de um registrador numa unidade de armazenamento (memória local), são agrupadas em campos (bits dentro da microinstrução). Num projeto de codificação indireta (dois níveis de codificação), o significado de um campo depende do valor de um outro campo na microinstrução. Numa estrutura de codificação em dois níveis pode-se ter também o significado de um campo dependente do estado do sistema indicado pelo conteúdo do registrador de estado.

Enquanto que a codificação reduz o tamanho da microinstrução e portanto o tamanho da memória de controle, são necessários mais circuitos e mais tempo para decodificar as microinstruções. Uma palavra de controle muito codificada pode tornar a microporgramação muito difícil.

Microinstruções verticais empregam tipicamente codificação em dois níveis. O código de operação indica a maneira como devem ser interpretados os outros campos. Nem toda organização vertical precisa utilizar codificação em dois níveis, podendo mesmo variar o nível de codificação de campo para campo dentro de uma mesma microinstrução.

- Implementação das microintruções

As microinstruções são executadas pelo controle numa maneira similar àquela em que são executadas as instruções de uma linguagem de máquina de uma máquina de controle fixo ("hardware"). A característica série-paralelo mede o quanto existe de sobreposição entre a execução da microinstrução corrente e a busca da próxima a ser executada.

Numa implementação série a busca da próxima microinstrução não começa até que a execução da microinstrução corrente termine. No outro extremo, a busca da próxima microinstrução a ser executada é feita em paralelo (simultaneamente) à execução da microinstrução corrente. A vantagem da busca sequencial é a sua simplicidade de realização; o "hardware" não precisa controlar simultaneamente execução e busca, e não existe nenhum problema na execução de pulos condicionais. A vantagem da busca paralela é a economia de tempo proporcionada pela sobreposição com a execução; a execução de uma microinstrução começa imediatamente depois de ter sido completada a execução da microinstrução anterior.

Quando não é possível ler-se a próxima palavra da memória de controle antes da execução da microinstrução corrente terminar, um esquema que combina a busca série e a paralela pode ser usado. Nesta situação quando o próximo endereço é conhecido no começo da execução da presente microinstrução, a próxima é lida em paralelo à execução. Quando o próximo endereço depende de condições cujo estado será estabelecido durante a execução da microinstrução corrente, o próximo endereço pode ser adivinhado e a respectiva palavra de controle é lida em paralelo com a execução. Somente se o endereço adivinhado estiver errado será necessário realizar uma nova leitura (série).

A característica monofásica-polifásica se refere ao número de fases (ciclos menores, subciclos) usadas para executar cada microinstrução. O conjunto de subciclos necessários à execução de uma microinstrução é chamado de ciclo completo ou simplesmente ciclo. Numa implementação monofásica,não há subciclos distintos do ciclo básico. Cada microinstrução é executada numa única emissão dos sinais de controle. Numa implementação polifásica cada ciclo completo compreende vários subciclos; o "hardware" gera os sinais de controle a cada subciclo.

A vantagem da implementação monofásica é a simplicidade de realização, quando não existe problema de corridas críticas. Operações polifásicas permitem maior interação entre as unida des do sistema, à custa de um controle mais complicado.

Uma vez definido o que vem a ser microprogramação, dentro do ponto de vista do projetista de sistemas digitais,e tendo sido já apresentados aspectos do projeto e implementa ção de microinstrução, ressaltando as suas principais carac terísticas, pode-se passar ao exame da estrutura de um sequen cializador com um núcelo de controle microprogramado.

- O núcleo de controle microprogramado

Através da definição de microprogramação, de imedia to percebe-se o quão flexível é um esquema de controle micro programado. Cria-se basicamente, um outro sistema digital que recebe comandos segundo uma linguagem pré-estabelecida (con junto de microinstruções) e a execução desses comandos (micro instruções) determinam operações em unidades funcionais do fluxo de dados. Ordenando-se as microinstruções de maneira conveniente (microprograma) pode-se implementar um determina- do algorítmo no sistema.

Sendo assim, a definição de uma estrutura apropria- da para o processador de microinstrução (microprocessador) po de vir precedida de um amplo estudo de arquitetura de siste mas. Mas seja qual for a estrutura escolhida, ela certamente se desenvolverá em torno de um esquema básico e fundamental.O ponto central de um sequencializador microprogramado é a memó ria de controle. Para ler as microinstruções da memória de controle, é necessário ter um registrador que forneça o ende reço da palavra de controle a ser lida.

Para se executar uma microinstrução é necessário colocá-la num registrador para poder decodificá-la e emitir todos os sinais de controle a ela associados. Para poder alterar condicionalmente a sequência de execução das microinstruções, existe o multiplexador de condições, que recebe variáveis de teste e indicadores de estado de todas as unidades funcionais do sistema, sendo comandado por campos especiais das microinstruções. Para separar os desvios de sequência provocados pelo próprio sequencializador da inicialização de uma sequência determinada pelo circuito identificador, existe o multiplexador de endereço da memória de controle.

Desta forma, as peças básicas e fundamentais de um sequencializador microprogramado podem ser organizadas da seguinte maneira: (Fig. II . 16).

Muitas estruturas diferentes podem ser definidas em torno deste esquema básico de núcleo de controle microprogramado. Várias outras unidades podem ser incluídas com a finalidade de se implementar outras características, tentando facilitar a microprogramação e a utilização eficiente dos recursos do "hardware" do sistema. Embora sendo muitas as alternativas de expansão, as funções básicas e essenciais para a implementação de um sequencializador microprogramado aparecem no esquema apresentado.

Somente a título de ilustração, pode-se notar que para implementar o conceito de microsubrotina, basta incluir mais um (ou vários) registrador no núcleo básico, com a finalidade de armazenar os endereços de retorno ao microprograma principal. Esta característica é de grande valia, pois possibilita economia em muitos trechos comuns de microprogramas. Uma vez estabelecido o núcleo de controle e a sua linguagem (conjunto de microinstruções), implementar um algorítmo no sistema é um simples problema de programação (microprogramação). Portanto, um núcleo de controle apresenta tanto características de "hardware" como de "software".

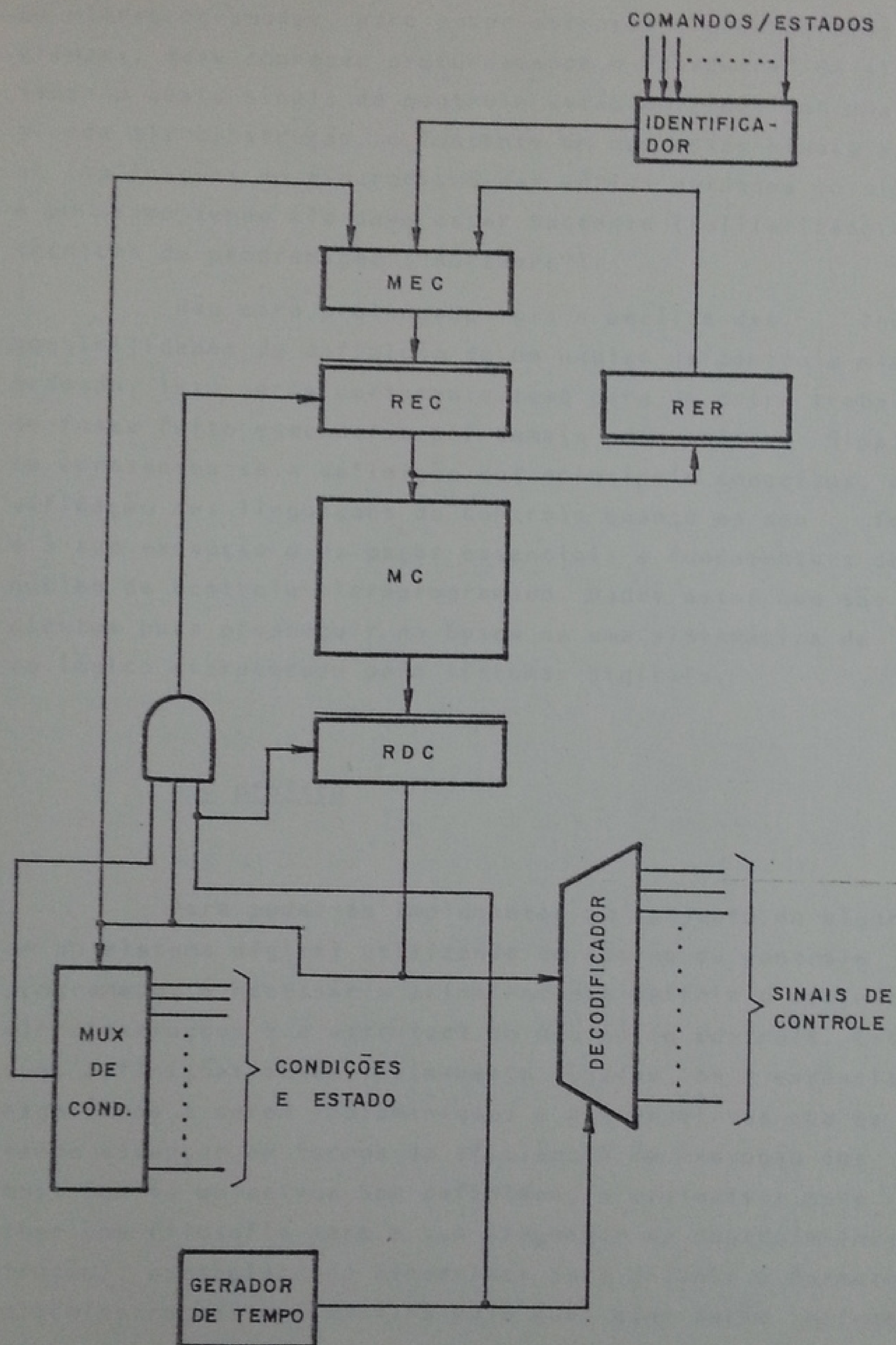


Fig.II.16. - Núcleo de Controle microprogramado

Um microprogramador, para poder escrever microprogramas eficientes, deve conhecer profundamente o "hardware" do sistema, sabendo quais sinais de controle serão emitidos por uma determinada microinstrução, o instante em que estes sinais atuam, as implicações no sincronismo das várias unidades do sistema, e ao mesmo tempo ele deve estar bastante familiarizado com as técnicas de programação ("software").

Não será prolongada mais a análise das inúmeras possibilidades de definição de um núcleo de controle microprogramado, isto seria certamente tema para um outro trabalho e se fosse feito estenderia por demais este relato. Simplesmente apresentou-se a definição dos principais conceitos, a classificação das linguagens de controle quanto ao seu formato e à sua execução e as peças essenciais e fundamentais de um núcleo de controle microprogramado. Dados estes que são suficientes para prosseguir na busca de uma sistemática de projeto lógico estruturado para sistemas digitais.

- <u>O projeto</u>

Para poder-se implementar um conjunto de algorítmos em um sistema digital utilizando um núcleo de controle microprogramado, é necessário primeiramente definir o conjunto de microinstruções e a estrutura do núcleo de controle. Estas duas definições estão intimamente ligadas com a essência dos algorítmos a serem implementados e dos objetivos que se pretende alcançar em termos da eficiência de execução dos mesmos. Com os objetivos bem definidos, o projetista pode escolher uma filosofia para a sua linguagem de controle (microinstrução), estabelecendo diretrizes para definir o formato das microinstruções e a maneira pela qual elas serão implementadas.

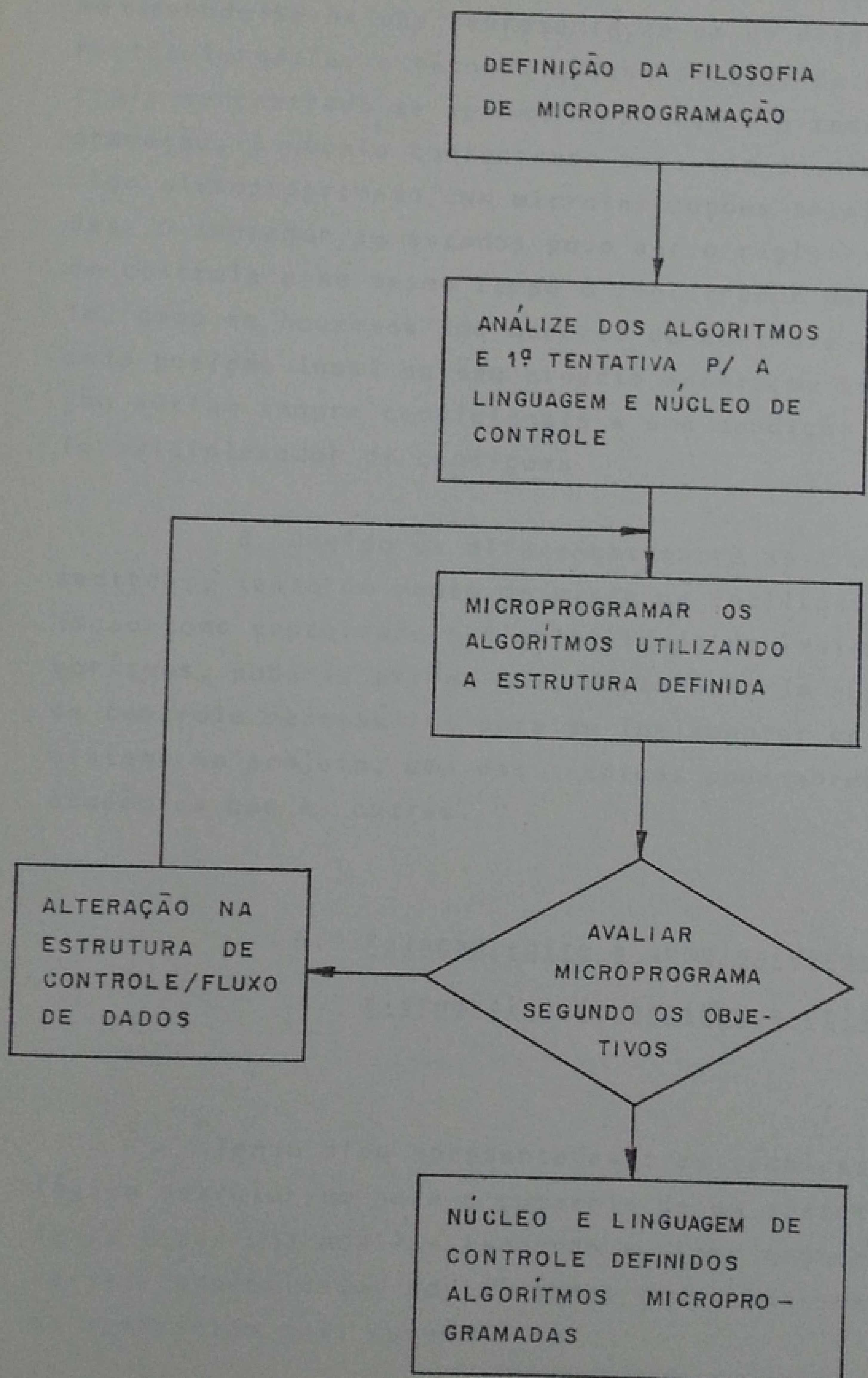
As alternativas possíveis para esta escolha já foram apresentadas nos dois sub-ítens anteriores.

Uma vez escolhida uma filosofia para a linguagem de controle, deve-se investigar minuciosamente os algorítmos a serem implementados, chegando até a identificar quais os subconjutnos de sinais de controle que comandam o fluxo de dados já definido, em cada uma das operações necessárias para a execução dos algorítmos. Desta análise, que não precisa ser exaustiva, pode-se perceber os conjuntos de operação que são mutuamente exlusivas. Fato este que permite um único campo da microinstrução, devidamente codificado, controlar um conjunto de sinais de controle mutuamente exclusivos. Após esta investigação prévia pode-se definir o formato das microinstruções e preparar então a linguagem de controle inicial (primeira aproximação).

Uma vez estabelecido uma primeira tentativa para as microinstruções e para o núcleo de controle, pode-se passar a microprogramação dos algorítmos a serem implementados utilizando a estrutura de controle proposta. A análise dos microprogramas obtidos pode sugerir uma série de alterações no formato e no conjunto de microinstruções, no núcleo de controle e até mesmo no fluxo de dados. Admitindo-se que algumas alterações são feitas (o que é bastante comum) após esta análise, deve-se microprogramar novamente os algorítmos e avaliá-los novamente, tendo como critério de avaliação os objetivos a serem alcançados. Este processo permanece nestas duas fases até que o projetista se dê por satisfeito com os resultados alcançados.

Esquematicamente a sistemática de projeto pode ser representada da maneira como mostra o diagrama ao lado.

Não será apresentado aqui um exemplo desta sistemática de projeto, uma vez que será descrito no capítulo IV o projeto do processador central, que é o objetivo deste trabalho.


DEFINIÇÃO DA FILOSOFIA DE MICROPROGRAMAÇÃO
ANÁLIZE DOS ALGORITMOS E 1º TENTATIVA P/ A LINGUAGEM E NÚCLEO DE CONTROLE
MICROPROGRAMAR OS ALGORÍTMOS UTILIZANDO A ESTRUTURA DEFINIDA
AVALIAR MICROPROGRAMA SEGUNDO OS OBJETIVOS
ALTERAÇÃO NA ESTRUTURA DE CONTROLE/FLUXO DE DADOS
NÚCLEO E LINGUAGEM DE CONTROLE DEFINIDOS ALGORÍTMOS MICROPROGRAMADAS

Observações:

A. É conveniente ressaltar a esta altura, a semelhança existente entre as tres técnicas de projeto lógico apresentadas. Todas se fundamentam na implementação de um algorítmo baseando-se na sua representação em um diagrama de blocos. Particularmente, a técnica que se utiliza de um núcleo de controle concentrado se assemelha bastante à técnica de microprogramação. O núcelo concentrado pode ser encarado como um núcleo microprogramado com microinstruções totalmente codificadas. O contador de estados pode ser o registrador de endereço de controle e ao mesmo tempo o registrador de dados de controle, como se houvesse uma memória de controle com o conteúdo de cada posição igual ao seu próprio endereço. Estas microinstrução seriam sempre condicionais a uma condição determinada pelo multiplexador de condições.

B. Devido às diferenças entre as tres técnicas apresentadas, tanto do ponto de vista de facilidade de implementação como capacidade e flexibilidade na implementação de algorítmos, pode-se prever que, dependendo do número de passos de controle necessários para se implementar os algorítmos do sistema em projeto, uma das técnicas pode apresentar-se mais econômica que as outras.

6. Relação entre a complexidade do sistema e a sistemática de projeto a utilizar.

Tendo sido apresentadas tres técnicas de projeto lógico estruturado para o controle de um sistema digital, pode-se fazer uma análise tentando estimar grosseiramente as diversas complexidades dos sistemas onde cada uma das técnicas se apresentam mais conveniente.

Para se fazer este tipo de análise deve-se recorrer a modelos econômicos dos tres núcleos de controle apresentados. Se esta análise for encarada com o maior rigor e exatidão possível, certamente não será encontrado nenhum modelo econômico viável para tal. Portanto, lembrando-se que o intuito é simplesmente ter uma ordem de grandeza da faixa de aplicação de cada uma das técnicas, relativamente à complexidade do sistema (número de passos de controle necessários para implementar os algorítmos), podemos nos aventurar neste delicado terreno. É importante ressaltar, antes de iniciar a análise, que não será levado em conta aspectos relativos à performance do sistema se implementado segundo uma técnica ou outra. Isto se justifica plenamente devido as modestas pretensões esperadas como resultado desta análise.

Para poder estabelecer uma comparação entre o controle distribuido, o controle concentrado e o microprogramado, deve-se escrever uma expressão matemática[12] para o custo do circuito resultante da aplicação de cada uma das técnicas num mesmo projeto.

Para escrever as expressões analíticas do custo, deve-se estabelecer a seguinte notação:

C_m : custo médio por módulo de "hardware"

C_p : custo médio de projeto para a inclusão de um módulo no sistema.

CW : custo de uma palavra de controle

C_M : custo total de um núcleo de controle microprogramado.

C_H : custo total de um núcleo de controle "hardwired" (concentrado ou distribuido).

Para implementar um certo algorítmo através de um núcleo "hardwired", supõe-se que seja necessário n_H módulos. Portanto o custo total do circuito resultante será:

$$C_H = (C_m + C_p)\ n_H \qquad (1)$$

Já para implementar esse mesmo algorítmo com um núcleo de controle microprogramado, deve-se ter um custo associado ao circuito de suporte para a memória de controle. Supondo que o circuito de suporte (unidade de decodificação, multiplexador de condição, unidades registradoras, etc...) seja composto por n_s módulos de "hardware", a expressão para seu custo será:

$$C_S = (C_m + C_p)\ n_S$$

Assumindo que o custo da memória de controle seja proporcional ao número de palavras de controle e que o número de palavras necessárias para microprogramar o algorítmo seja N_I, tem-se para o custo da memória de controle a seguinte expressão:

$$C_{MC} = C_W\ n_I$$

Assumindo também que o custo para microprogramar e codificar uma palavra de controle seja C_c, portanto, o custo de microprogramação e codificação dosalgorítmos seria:

$$C_A = C_c\ n_I$$

Com o custo já calculado das diversas partes que compõem genericamente um núcleo de controle microprogramado, pode-se escrever a expressão analítica que dá o custo total de uma unidade de controle microprogramada:

$$C_M = C_S + C_{MC}\ C_A$$

ou substituindo

$$C_M = (C_m + C_p)\ n_S + C_w\ n_I + C_c\ n_I \qquad (2)$$

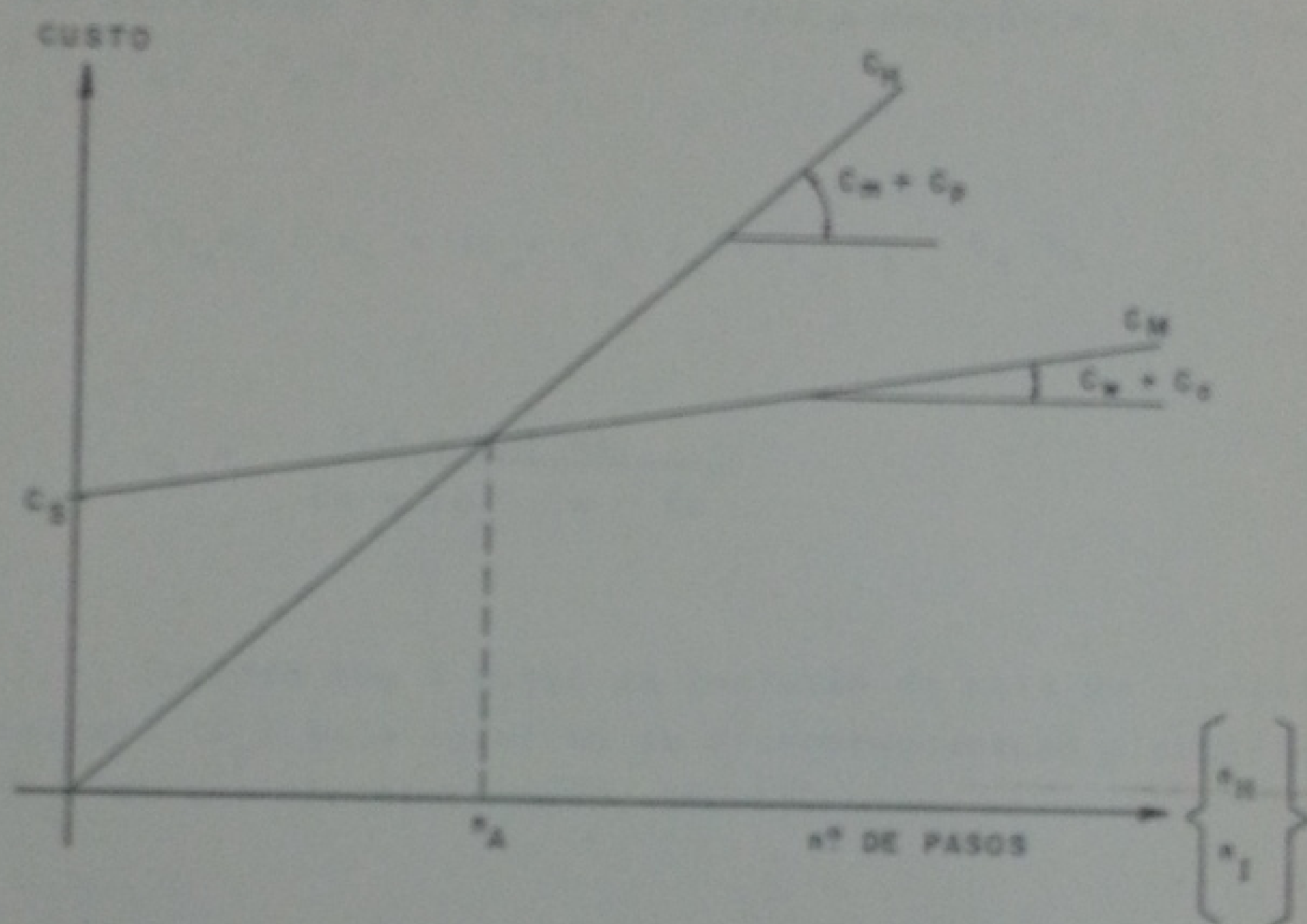
Através desta expressão pode-se observar que para um núcleo de controle onde não se implementa nenhum algoritmo, ou seja, não possui memória de controle ($n_I = 0$), ainda assim tem-se um custo inicial C_S.

Com a inclusão da memória de controle, o custo da unidade de controle aumenta proporcionalmente à complexidade do sistema (n_I) na razão de ($C_w + C_c$). Razão esta que é sensivelmente menor que ($C_m + C_p$), razão de crescimento do custo de um núcleo "hardwired". (uma porção de memória de controle de 128x8 bits custa bem menos que 128 módulos de "hardware" MSI, e os custos de projeto C_c e C_p podem ser considerados aproximadamente iguais).

Com as expressões obtidas e as considerações feitas pode-se esboçar um gráfico custo X número de passos de controle (complexidade do sistema): (figura ao lado).

O problema de identificação da faixa de aplicação das sistemáticas de projeto se resume na determinação do ponto de intersecção das curvas C_H e C_M. Observando o gráfico acima pode-se retirar as seguintes conclusões:

- para sistemas com um número de passos de controle inferior a n_A é mais conveniente a aplicação das técnicas "hardwired" (núcleo distribuido ou concentrado).


CUSTO
c_M
c_S
n_A
nº DE PASOS

- para sistemas onde o número de passos de controle é superior a n_A é mais conveniente a aplicação de um núcleo de controle microprogramado.

Com as expressões obtidas pode-se isolar a variável n_A; das expressões 1 e 2 pode-se tirar a equação de n_A, fazendo-se $C_H(n_A) = C_M(n_A)$ (3)

$$(C_m + C_p)\ n_A = (C_m + C_p)\ n_S + C_w\ n_A + C_A\ n_A$$

$$n_A = \frac{(Cm + Cp)\ ns}{Cm + Cp - Cw - Cc}$$

Supondo que o custo de inclusão de mais um módulo de controle (C_p) seja igual ao de microprogramação e codificação (C_c) de uma microinstrução, temos:

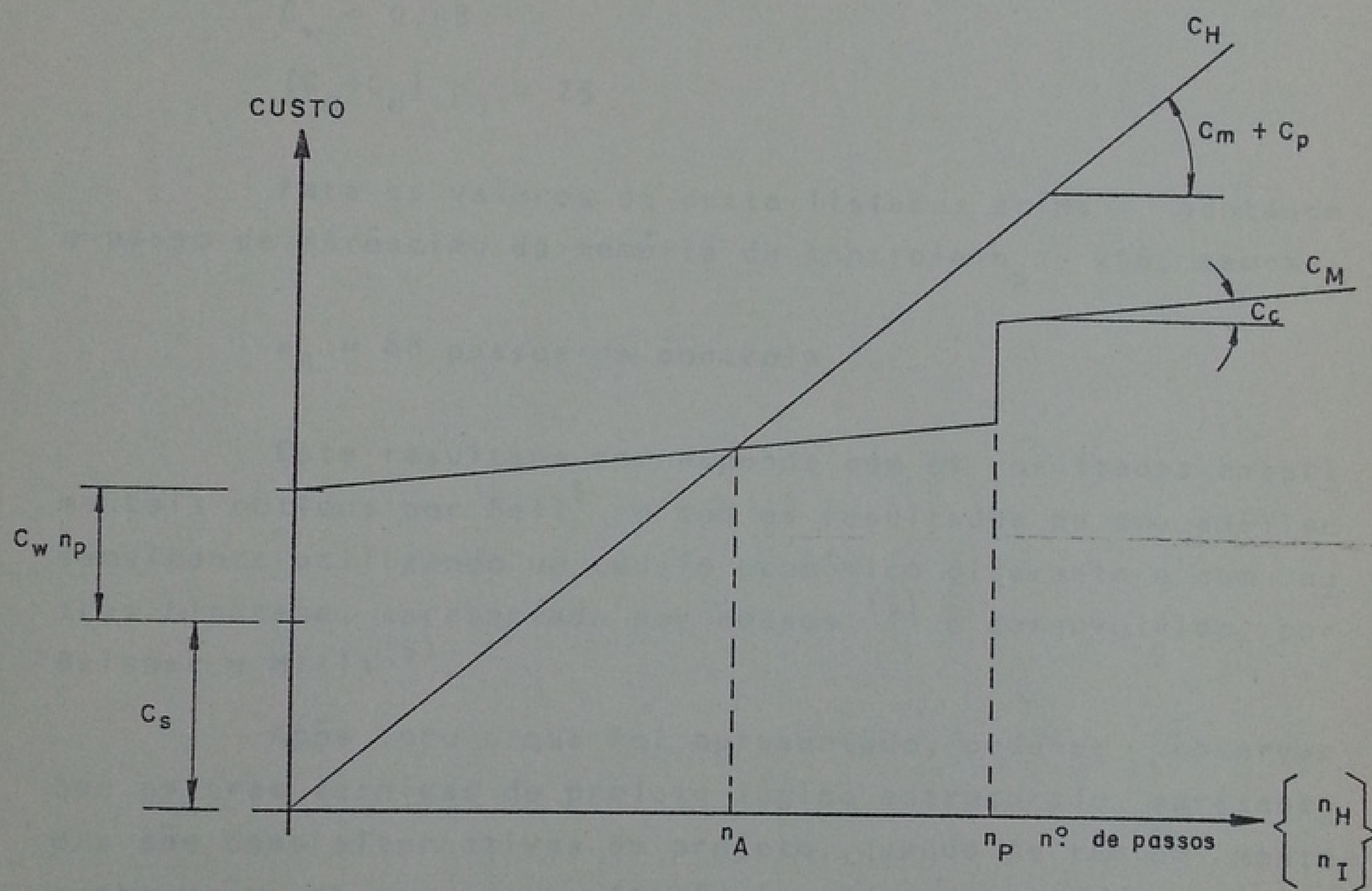
$$n_A = \frac{(Cm + Cp)\ n_s}{Cm - Cw}$$

Para chegar a esta expressão de n_A foi suposto que a memória de controle crescesse em incrementos de uma palavra, o que na realidade não é verdade, em geral este incremento é igual a um número bem maior que 1. Isto é devido aos módulos MSI de memória de controle disponívies atualmente. Portanto se for chamado de n_p o passo de crescimento da memória de controle, a equação 2 fica sendo:

$$C_M = (C_m + C_d)\ n_s + C_w\ n_p + C_c\ n_I$$

onde o custo inicial do núcleo de controle microprogramado é dado por: $(C_m + C_d)\, n_s + C_w\, n_p$

A representação gráfica anteriormente feita toma o seguinte aspecto:



Portanto pode-se chegar a uma nova expressão para n_A:

$$n_A = \frac{(C_m + C_d)\, n_s + C_w\, n_p}{Cm} \qquad (5)$$

Para poder estimar grosseiramente a ordem de grandeza de n_A, serão utilizados valores típicos para as variáveis que aparecem na equação 5, que podem ser encontradas na referência 6, representados numa unidade proporcional ao preço de venda (1972) dos respectivos módulos. Estes valores são os seguintes:

$$C_m = 0,54$$

$$C_w = 0,08$$

$$(C_m + C_d)\ n_S = 25$$

Para os valores de custo listados acima e adotando o passo de acréscimo da memória de controle $n_p = 256$, tem-se:

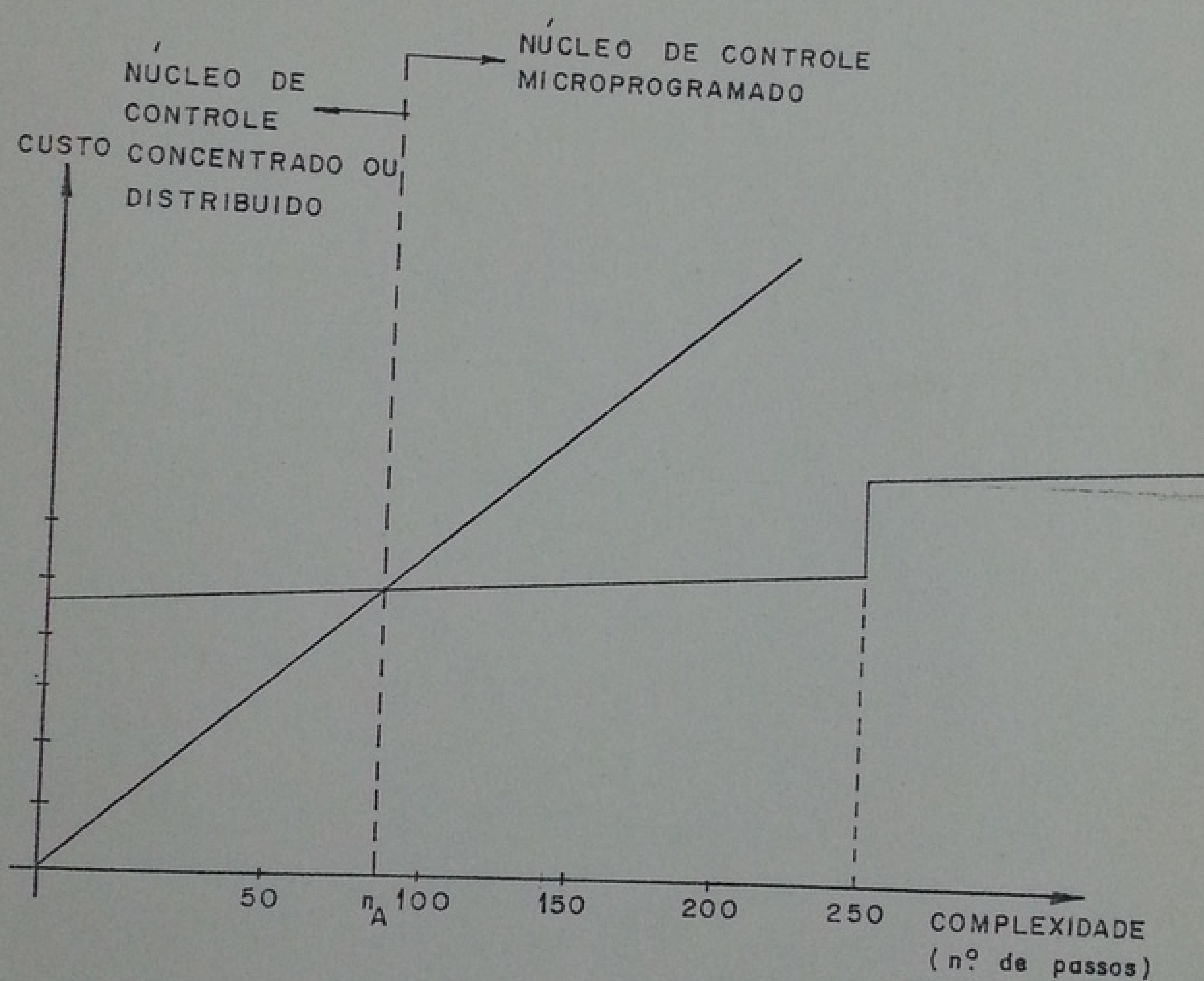
$$n_A = 88 \text{ passos de controle}$$

Este resultado corresponde com os resultados experimentais obtidos por Bell[6] e com os resultados de uma análise semelhante utilizando um modelo econômico diferente e com outras hipótese, apresentado por Husson [(5)] e desenvolvido por Beisner e Mills[(5)].

Após tudo o que foi apresentado, pode-se observar que as tres técnicas de projeto lógico estruturado apresentadas são boas alternativas de projeto, quando se tem em mente a obtenção de um sistema flexível, modular, que utilize eficientemente as facilidades disponíveis através dos circuitos integrados MSI, que seja facilmente projetado, depurado, entendido, documentado e mantido em funcionamento (manutenção).

Ainda deve-se notar que para sistemas digitais cuja complexidade é pequena, as técnicas de controle concentrado e distribuido se tornam mais recomendáveis que o controle microprogramado. Conforme o sistema vai crescendo em complexidade a técnica de controle microprogramado vai se tornando mais conveniente que o controle concentrado ou distribuido.

Tendo sido estimado o grau de complexidade do sistema (n_A), a partir do qual o núcleo de controle microprogramado se torna mais conveniente para o projeto do controle de sistemas digitais, pode-se representar estas conclusões, esquematicamente, pelo seguinte gráfico:



Baseando-se nestas conclusões e observando-se na descrição da arquitetura do processador, a complexidade e o volume dos algorítmos a serem implementados, optou-se pelo projeto estruturado de um processador microprogramado, cuja estrutura será descrita no capítulo IV deste trabalho.

Baseando-se nestas conclusões e observando-se na descrição da arquitetura do processador, a complexidade e o volume dos algorítmos a serem implementados, optou-se pelo projeto estruturado de um processador microprogramado, cuja estrutura será descrita no capítulo IV deste trabalho.

III. O SISTEMA MODULARIZADO EM NÍVEL PMS

Tendo sido apresentados e esclarecidos os principais objetivos a serem alcançados neste projeto e estabelecidas algumas regras e sistemáticas para a obtenção dos mesmos, pode-se passar à descrição das soluções adotadas.

Primeiramente, será descrita a estrutura escolhida para prover o sistema com um alto grau de modularidade em nível PMS. Serão apresentados os módulos PMS utilizados e o protocolo de comunicação entre esses blocos. Após esta descrição global do sistema, no capítulo IV será discutido somente o processador central, descrevendo tanto o seu fluxo de dados como o controle, evidenciando os aspectos principais de um projeto lógico estruturado (modularizado em nível RT).

Para o completo entendimento das soluções a serem apresentadas, supõe-se pelo menos um conhecimento superficial da arquitetura do processador central, que aparece descrita de maneira resumida no apêndice deste trabalho.

1. O sistema

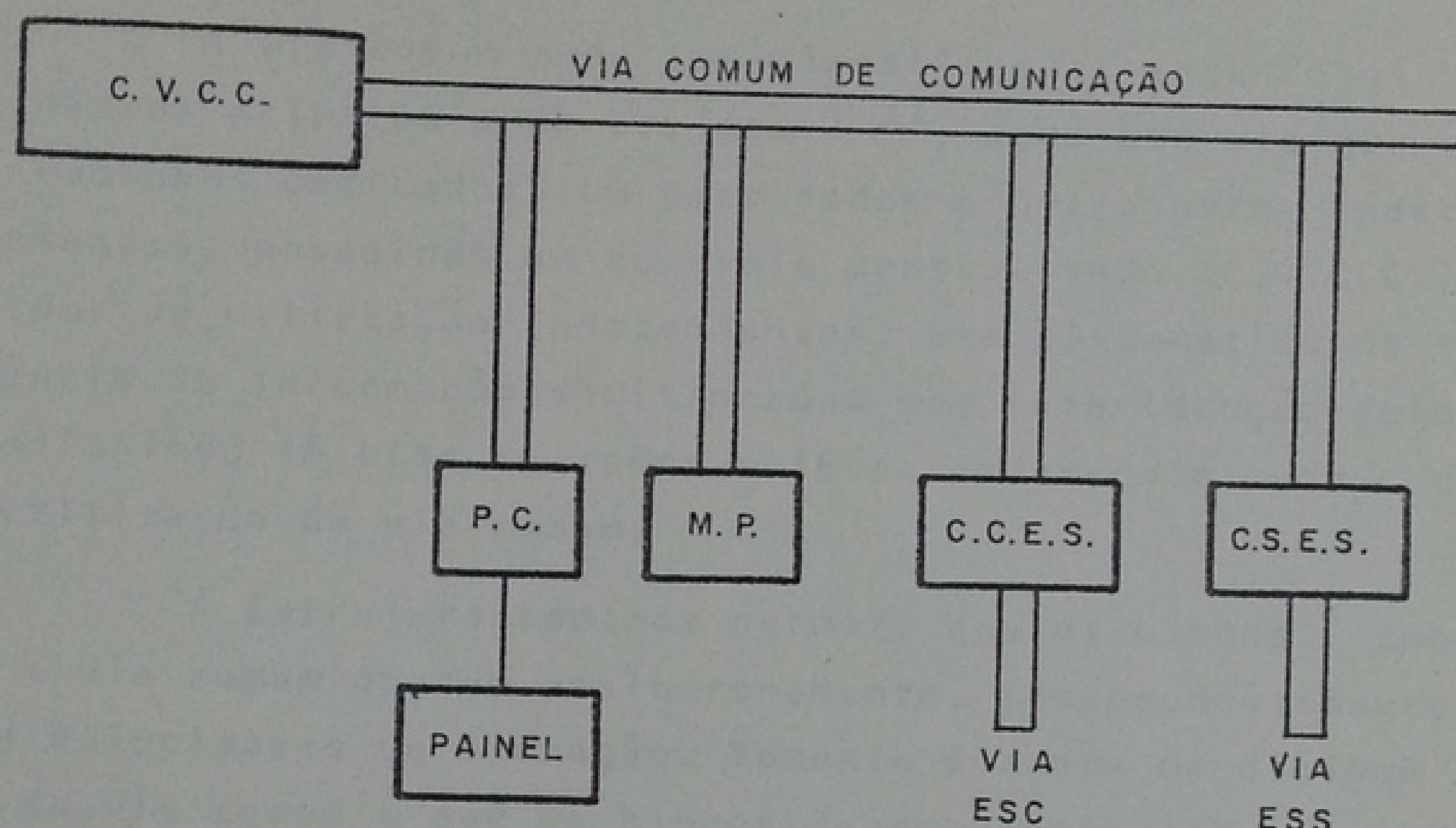
O sistema digital em projeto baseia seu funcionamento em uma VIA Comum de Comunicação, à qual estão conectados os diversos módulos que realizam as funções básicas do sistema, e através da qual tais blocos interagem de maneira assíncrona. Tal sistema é apresentado na figura III.1 na qual podem ser vistos os quatro tipos básicos de blocos definidos a seguir:

- BLOCO DE PROCESSAMENTO: - É o bloco onde se realiza o processamento central da máquina. A distribuição e gerenciamento dos outros recursos do sistema são feitos por programas nele executados. Dá-se a este bloco o nome de Processador Central (P.C).

- BLOCO DE ARMAZENAMENTO: - É o bloco de armazenamento primário do sistema. Todos os programas e seus respectivos dados deverão estar armazenados no bloco de armazenamento para serem executados. Dá-se a este bloco o nome de Memória Primária (M.P).

- BLOCO DE ENTRADA E SAÍDA CONCENTRADOR: - É um dos blocos que monitora as operações de entrada e saída de dados entre a Unidade de Processamento e os Dispositivos Periféricos, com capacidade de controlar diversos deles ao mesmo tempo. Pela característica de concentração, este tipo de bloco será utilizado no monitoramento de uma via de entrada e saída onde serão conectados dispositivos periféricos de baixa velocidade. Dá-se a este tipo de bloco o nome de Canal Concentrador de Entrada e Saída (C.C.E.S.).

- BLOCO DE ENTRADA E SAÍDA SELETOR: - É um dos blocos que monitora as operações de entrada e saída de dados no sistema, operando com um dispositivo periférico por vez. Inicializada uma tarefa de entrada ou saída através deste bloco, somente após o término desta operação, é que se pode proceder à inicialização de uma nova tarefa. Devido a esta característica seletiva, este bloco pode controlar uma via de entrada e saída onde serão conectados dispositivos periféricos de alta velocidade. Dá-se a este bloco o nome de Canal Seletor de Entrada e Saída (C.S:E.S.).



- VIA ESC : - VIA DE ENTRADA E SAÍDA CONCENTRADORA (BAIXA VELOCIDADE)
- VIA ESS : - VIA DE ENTRADA E SAÍDA SELETORA (ALTA VELOCIDADE)

Fig. III.1. Estrutura do Sistema

A troca de informações entre os diversos blocos que irão compor o sistema, será feita através da Via Comum de Comunicações, partilhada por todos eles, sendo que somente dois blocos podem, ao mesmo tempo, ser por ela conectados. A escolha e o monitoramento dos blocos que, em determinado instante, comunicam-se através da via comum é feito por uma unidade

controladora, o Controle da Via Comum de Comunicação, que além das funções já mencionadas decide sobre as prioridades de utilização da Via em situações de conflito, quando existem dois ou mais pedidos de uso da via.

A via comum pode ser classificada, segundo os seus parâmetros principais de projeto (13), como uma via que possui caminhos dedicados, um para dados e outro para endereços e comandos, possuindo um controle centralizado (C.V.C.C.) com pedidos de utilização independentes, uma sistemática de transferência de informação semisíncrona sem interlocução total, transferindo 16 bits de dados e 16 de endereço a cada ciclo de utilização da via comum.

A estrutura adotada permite que os blocos conectados à via comum operem assincronamente, independentemente de suas velocidades de operação. Somente durante um diálogo através da via comum é que os blocos devem operar sob controle do C.V.C.C. Sendo assim, o sistema não vincula a velocidade de cada um dos blocos. Por exemplo, pode-se conectar blocos de armazenamento que operem memórias de diferentes velocidades (núcleo ou semi condutor) sem nenhuma dificuldade. O mesmo acontece com os blocos de entrada e saída.

Todo bloco conectado à Via Comum deve possui um circuito de interfaceamento com o C.V.C.C. que gera os sinais necessários para a realização dos diversos diálogos, tornando possível a ligação de blocos independentemente de suas características. Dá-se a este circuito o nome de Interface com a Via Comum (I.V.C.).

2. A Via Comum

2.1. Definições

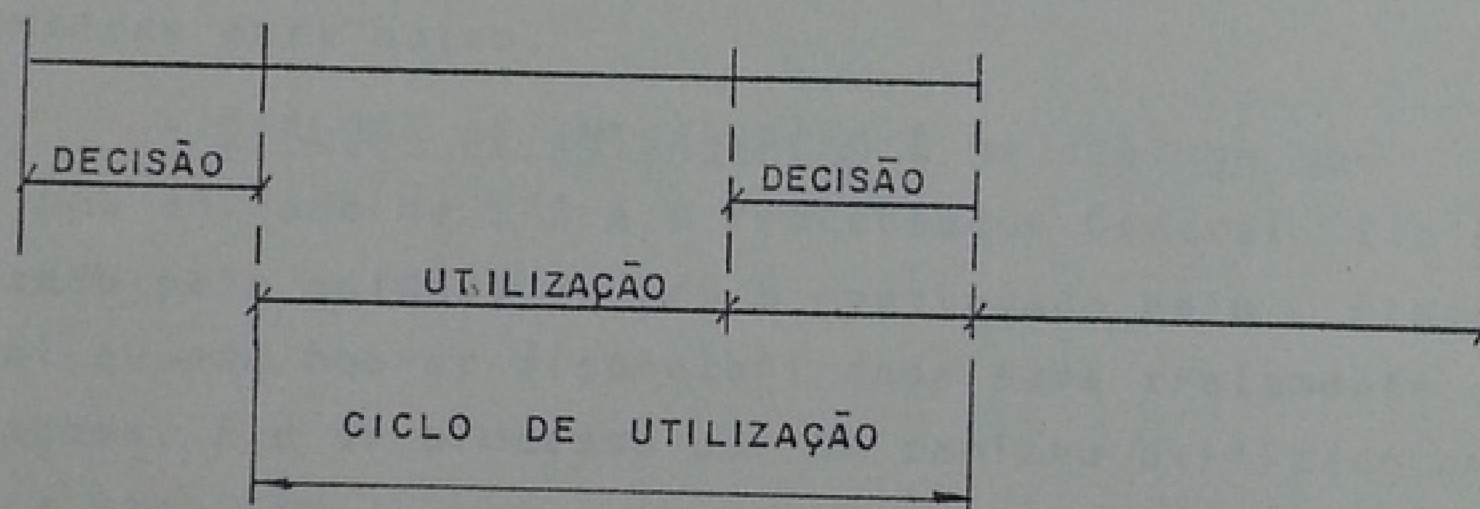
UNIDADE ATIVA:-É toda unidade que está conectada à via comum com direito de requisitar outras unidades do sistema.

UNIDADE PASSIVA:-É toda unidade que está conectada à via comum que pode ser requisitada por uma outra unidade ativa.

CICLO DE UTILIZAÇÃO DA VIA COMUM:-É o intervalo de tempo entre o início e o término de uma transação entre a unidade Ativa e Passiva, na via comum.

Uma comunicação através da via comum compõe-se basicamente de duas fases: Fase de Decisão e Fase de Utilização:

Na fase de Decisão é escolhida a unidade que será ativa no próximo ciclo da via e na Fase de Utilização um "diálogo" é estabelecido.



A duração da fase de utilização é variável, dependendo da tarefa a serrealizada.

Tem-se uma duração da ordem de 80ns para a fase de decisão e da ordem de 240ns, no caso mínimo, para a fase de utilização.

DIÁLOGO:-É o conjunto de sinais trocados por unidades ativas e passivas, durante um ciclo da via comum.

As unidades que serão conectadas à via comum poderão ser: somente Ativas, somente Passivas, ou Ativas e Passivas. A cada ciclo o controlador nomeia uma entre as unidades que no momento estão requisitando a via para ser Ativa e esta por sua vez escolhe a unidade com a qual pretende comunicar-se. Existem vários tipos de diálogos possíveis entre unidades ativas e passivas, são eles:

- DIÁLOGO DE PRIORIDADE: É aquele entre uma unidade Ativa e o controlador da via comum. Este diálogo se processa enquanto a via está em plena utilização, e seu objetivo é determinar a próxima unidade a utilizá-la.

- DIÁLOGO DE UTILIZAÇÃO: É o diálogo que se processa entre uma unidade Ativa e a Unidade Passiva requisitada. É através dele que se realiza a transferência de informações de uma unidade para outra.

- DIÁLOGO DE INTERRUPÇÃO:É um diálogo que ocorre entre uma unidade de E/S e o Processador Central. Ele é inicializado pela unidade de E/S e continuado pelo Processador Central quando houver disponibilidade para tratamento de interrupçoes. Ele se processa por um caminho utilizado unicamente para esse fim.

2.2 Controlador da Via Comum de Comunicação (C.V.C.C.)

O C.V.C.C. é a unidade que monitora a utilização da VIA COMUM. Esta unidade recebe dos diversos blocos do Sistema as requisições de utilização da VIA e decide qual o bloco de maior prioridade que a utilizará.

Neste sentido pode-se ter no sistema blocos que podem requisitar a VIA COMUM e blocos que não podem fazê-lo. São as unidades Ativas e Passivas, já definidas anteriormente. Os blocos do sistema, de acordo com este critério, são classificados como a seguir:

- MEMÓRIA PRIMÁRIA:- Bloco "somente Passivo"

- PROCESSADOR CENTRAL:- Bloco "Ativo"

- CANAL CONCENTRADOR:- Bloco "somente Passivo"

- CANAL SELETOR:- Bloco "Ativo e Passivo"

O controlador pode ser dividido basicamente em duas partes: Circuito de Prioridade e Circuito de Sincronismo. A fase de decisão é monitorada pelo Circuito de Prioridades enquanto que a de utilização é monitorada pelo de Sincronismo.

Para um bloco conseguir usar a VIA COMUM, ele deve enviar um sinal de pedido ao controlador. Este sinal excitará o circuito de prioridade que enviará um sinal resposta ao bloco prioritário. O sinal resposta colocará na VIA as informações de dados, controle e endereço do bloco Ativo neste ciclo. Com as linhas de endereço da VIA alimentadas pelo bloco Ativo, será selecionado o bloco Passivo que participará do diálogo de utilização. Todos os blocos conectados à VIA possuem um endereço através do qual eles são selecionados. Quando um bloco é selecionado ele se torna Passivo posicionando-se de acordo com as informações já colocadas na VIA pelo bloco Ativo.

O diálogo de utilização, monitorado pelo controlador, é iniciado em sincronismo com a base de tempo do mesmo, através do sinal "INÍCIO". Desta forma o bloco Passivo recebe as informações necessárias para realizar a tarefa pedida pelo bloco Ativo.

Durante um diálogo de utilização pode-se ter um ciclo cuja duração é variável.Dependendo do sentido da transferência a ser realizada, o controlador pode finalizar o ciclo sem esperar nenhuma sinalização vinda da unidade passiva (ciclo curto), ou se o sentido da transferência indicar um ciclo longo o controlador só finalizará este diálogo quando receber do bloco Passivo o sinal "PRONTO", que indica tarefa terminada no Passivo. A finalização do diálogo em ambos os casos ocorre em sincronismo com a base de tempo do controlador, através do sinal "FIM" enviado ao bloco Ativo. O sinal "FIM" inicia uma nova fase de decisão, transfere informações do Passivo para o Ativo, se necessário, e retira o sinal resposta do bloco Ativo liberando a VIA para um novo ciclo de utilização.

Quando a unidade passiva requisitada for a memória primária, o controlador verifica se ela não está ocupada. Se a memória estiver ocupada o controlador passa para uma nova fase de decisão ao invés de iniciar a fase de utilização. Permitindo assim, uma decisão mais justa e não congestionando a VIA.

O controlador opera com uma base de tempo de 12,5MHz 80ns. Ele inicia o cílco em sincronismo com este relógio e deve finalizá-lo também em sincronismo; portanto, quando ele recebe o aviso da unidade passiva para finalizar o ciclo de utilização ele só envia o sinal que encerra a presente utilização em correspondência com seu relógio central.

Sendo assim, todo ciclo da via comum deve ser múltiplo inteiro de 80ns. O ciclo mínimo é de 240ns, portanto, pode-se chegar a uma expressão que fornece as possíveis durações do ciclo da via comum neste esquema.

$$\text{CICLO} = 80\ (n + 3\ )\ \text{ns}, \quad n = 0,\ 1,\ 2,\ \ldots\ldots\ldots\ldots$$

2.3. Sinais da Via Comum

Temos quatro tipos de sinais pertencentes à Via Comum:

- SINAIS DE INFORMAÇÃO
- SINAIS DE CONTROLE
- SINAIS DE SINCRONISMO
- SINAIS DE INTERRUPÇÃO

SINAIS DE INFORMAÇÃO: São 35 sinais que carregam a informação que se quer transferir de uma unidade para outra. Eles são divididos em:

- DADOS: - 16
- ENDEREÇO: - 16 fios
- ENDEREÇO DE MÓDULO: - 3 fios

Os fios que conduzem os sinais de endereço são também utilizados para transferência de comandos de uma unidade para outra.

SINAIS DE CONTROLE: Sinais que estabelecem e monitoram a transferência de informações na Via Comum.

- PEDIDO - É o sinal pelo qual uma unidade requisita um ciclo de utilização da via comum. Este sinal é utilizado no diálogo de prioridade.

- RESPOSTA - É o sinal de resposta do controlador ao sinal de pedido. Seu objetivo é abrir as portas da unidade que foi nomeada ativa no presente ciclo. Este sinal é utilizado no diálogo de prioridade.

- SENTIDO - É o sinal que determina em que sentido devem ser abertas as portas de Informação das unidades a serem conectadas. Este sinal é utilizado no diálogo de utilização.

- PRONTO - É um sinal da unidade passiva para o controlador, avisando-o que a tarefa está executada. Este sinal é utilizado no diálogo de utilização.

SINAIS DE SINCRONISMO: Sinais que inicializam e finalizam um ciclo da via comum. Estes dois sinais são utilizados nos diálogos de utilização.

- INÍCIO - É o sinal que inicia a transferência de informação num diálogo de utilização.

- FIM - É o sinal que transfere uma informação da unidade passiva para a unidade ativa quando houver necessidade, e finaliza o ciclo de utilização da VIA.

SINAIS DE INTERRUPÇÃO: Sinais trocados entre os blocos de E/S e o Processador Central num diálogo de interrupção.

- PEDIDO DE INTERRUPÇÃO - Sinal enviado por um bloco de E/S para interromper o Processador Central.

- RECONHECIMENTO - Sinal enviado pelo Processador Central aos blocos de E/S para comunicar o recebimento da interrupção.

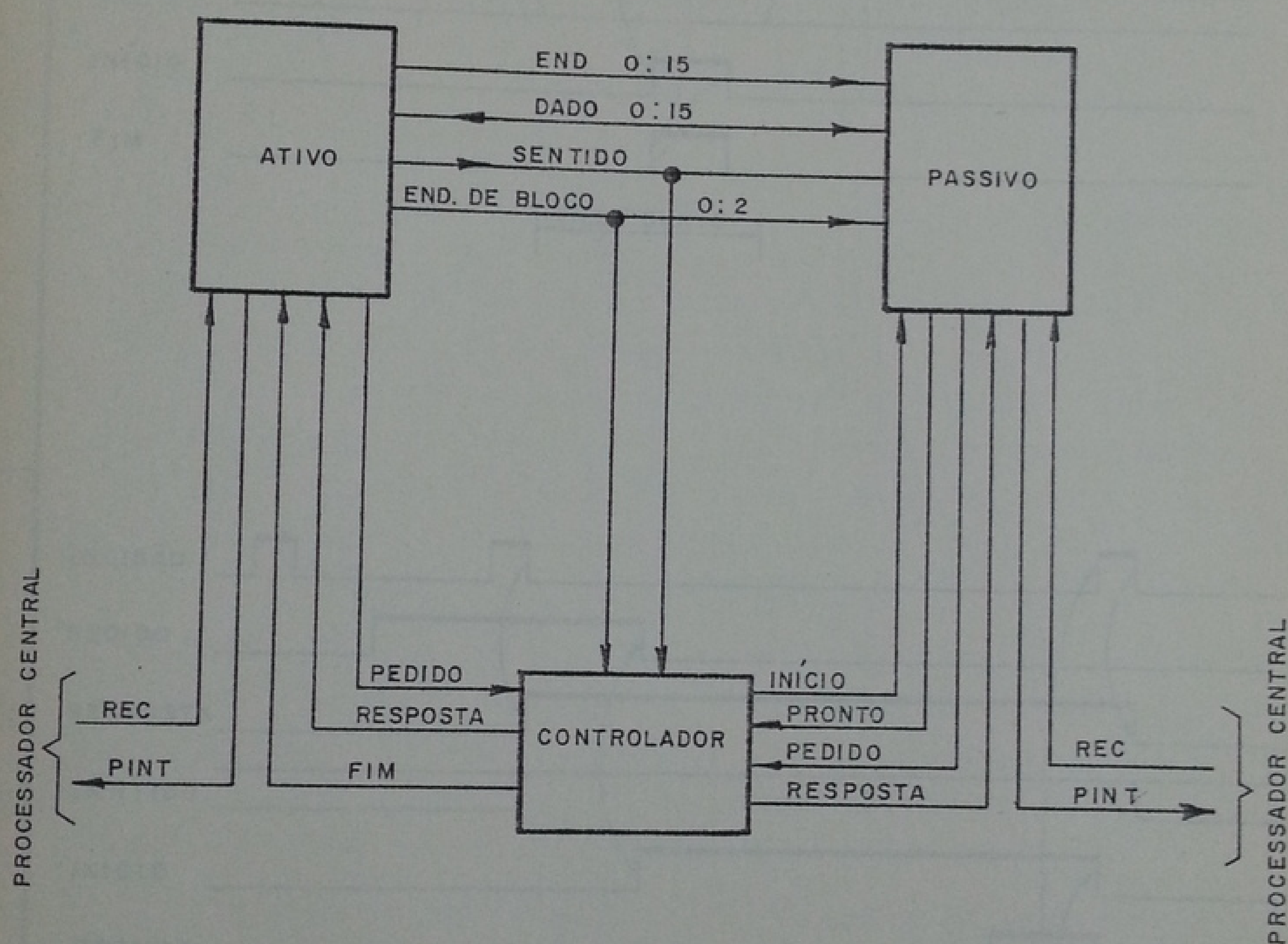
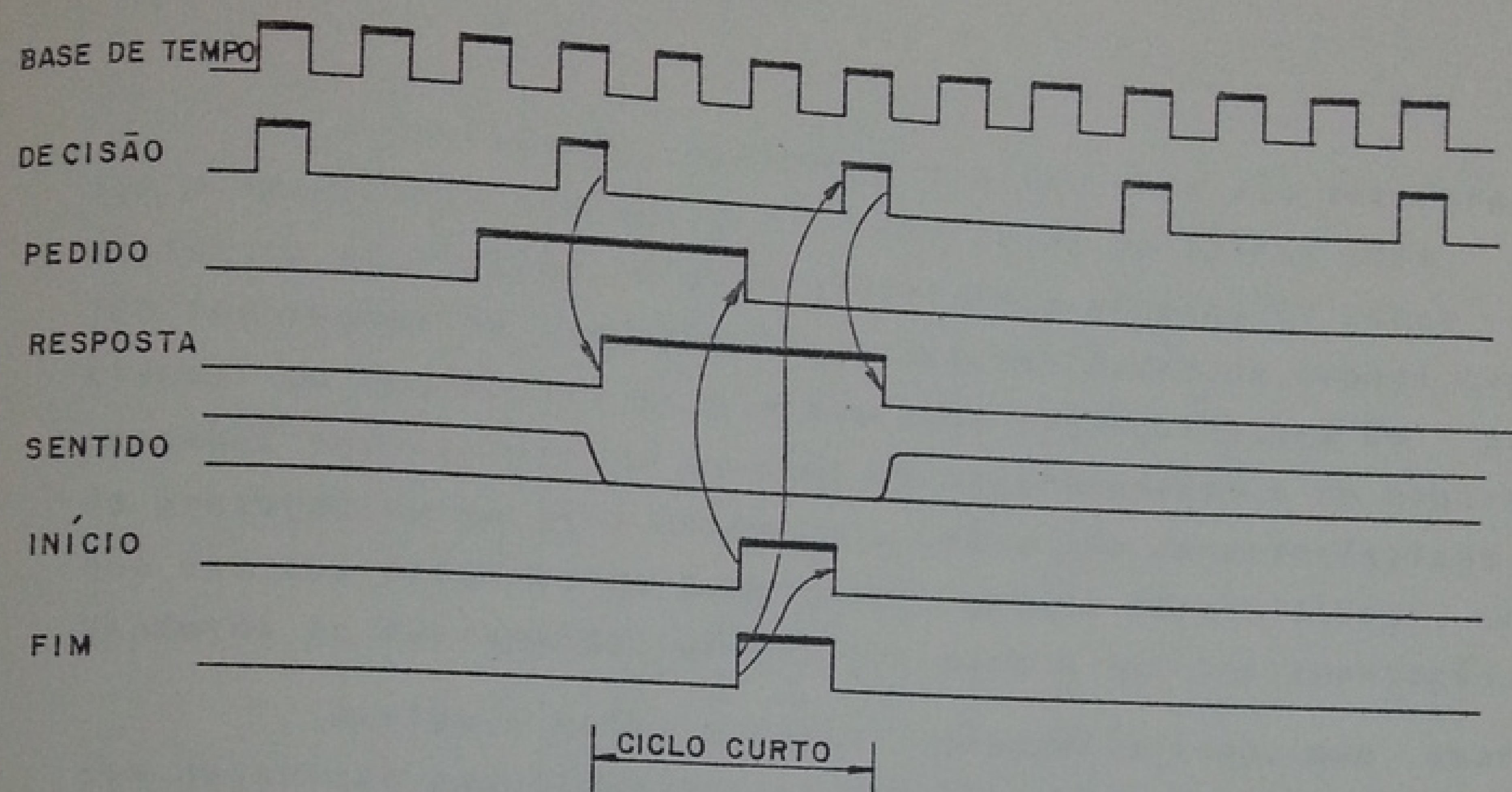


Fig. III.2. Sinais da Via Comum



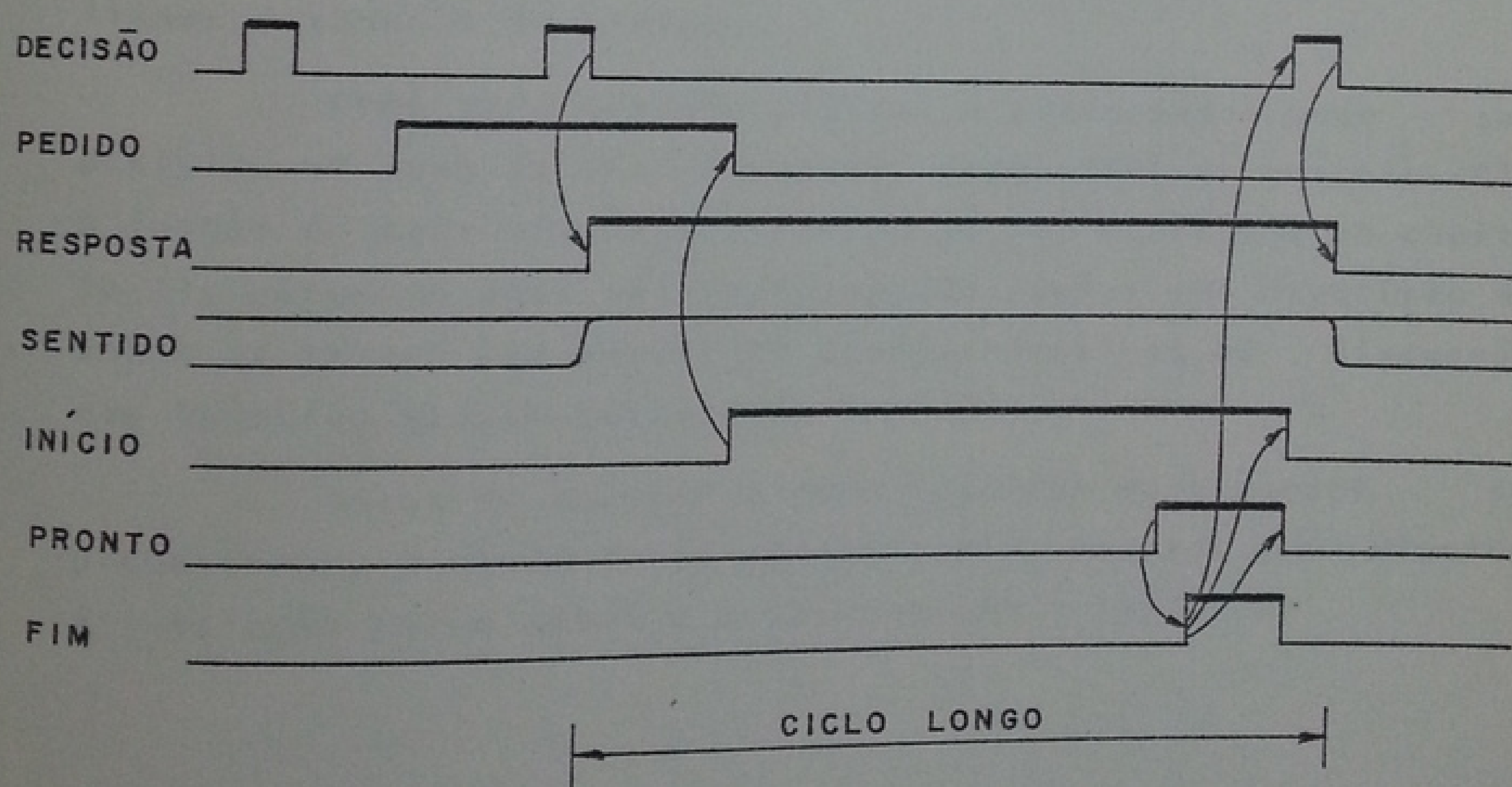


Fig.III.3. - Diagrama de tempo da via comum para um ciclo curto (saída) e um ciclo longo (entrada).

IV. O PROCESSADOR

O processador central é o bloco onde são interpretadas e executadas a maioria das instruções definidas pela arquitetura do sistema. O processador deve distinguir cerca de 120 instruções de máquina, sendo algumas delas de grande capacidade. Um dos pontos mais elaborados da arquitetura do processador encontra-se no esquema de endereçamento e no esquema de proteção da memória principal. Estas são características que dão aos sistemas de programação grande flexibilidade, provendo-os de ferramentas eficientes para a sua implementação.

Devido a estas e outras características, que aparecem descritas resumidamente no apêndice, é que o número de algorítmos a serem executados pelo processador é bastante grande, sendo alguns deles de grande complexidade. Desta forma foi necessário um minucioso estudo desses algorítmos, para estruturar de maneira eficiente os circuitos do processador. Levou-se em conta também objetivos tais como velocidade, expandibilidade e tamanho do sistema.

Como todo sistema digital o processador pode ser dividido em duas partes: Fluxo de dados (FD) e Controle (C). A função de cada uma dessas partes já foi analisada no capitulo II, portanto aqui será apresentada apenas uma descrição de ambas as partes, sem entrar em grandes detalhes na discussão das inúmeras alternativas para o projeto.

Antes de iniciar a descrição das duas partes do processador, é conveniente apresentar um esquema que elucide a interação entre ambas e a sua conexão à via comum.

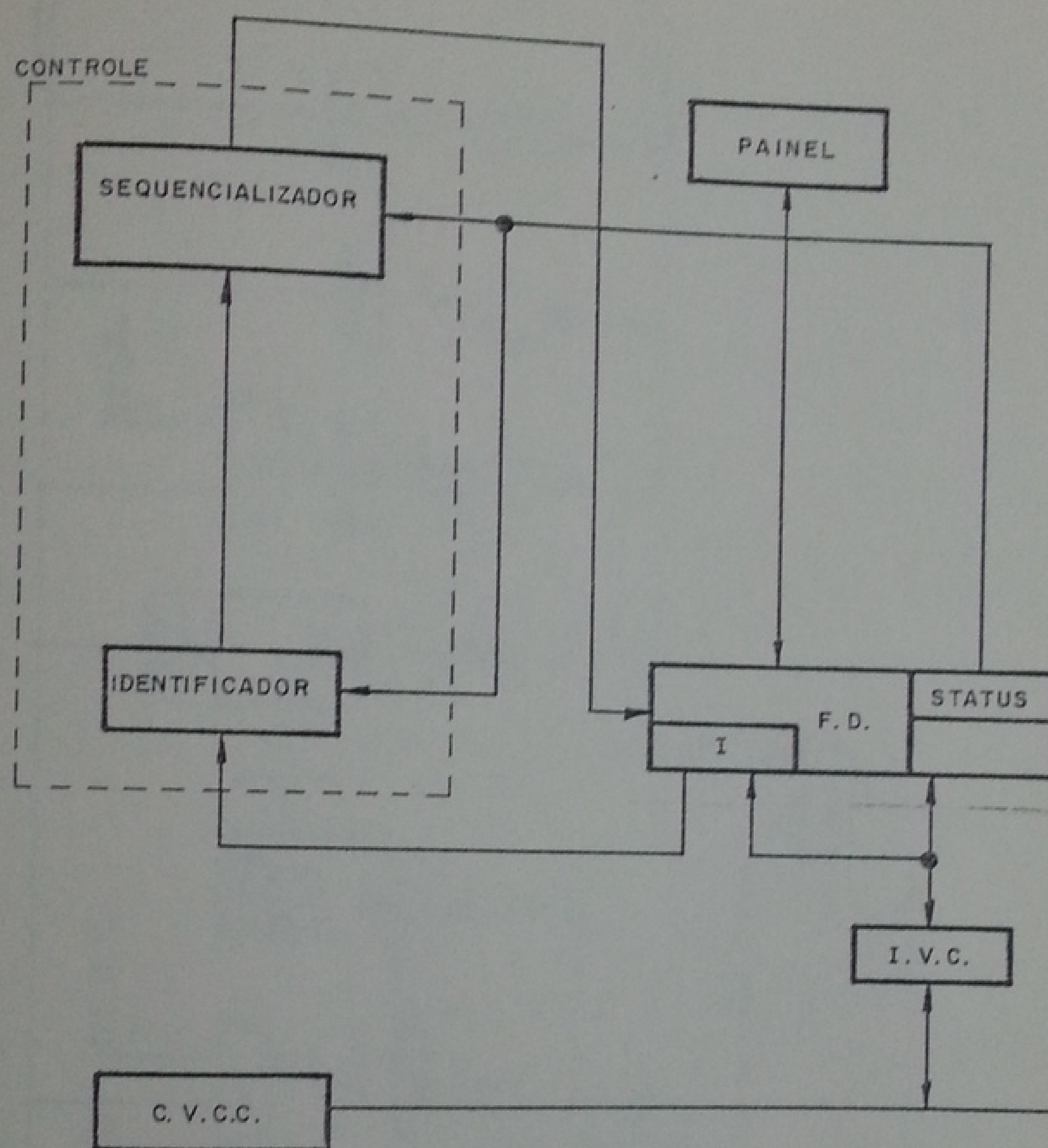


Fig. IV.1. Diagrama simplificado e resumido evidenciando o Fluxo de Dados e o Controle do Processador.

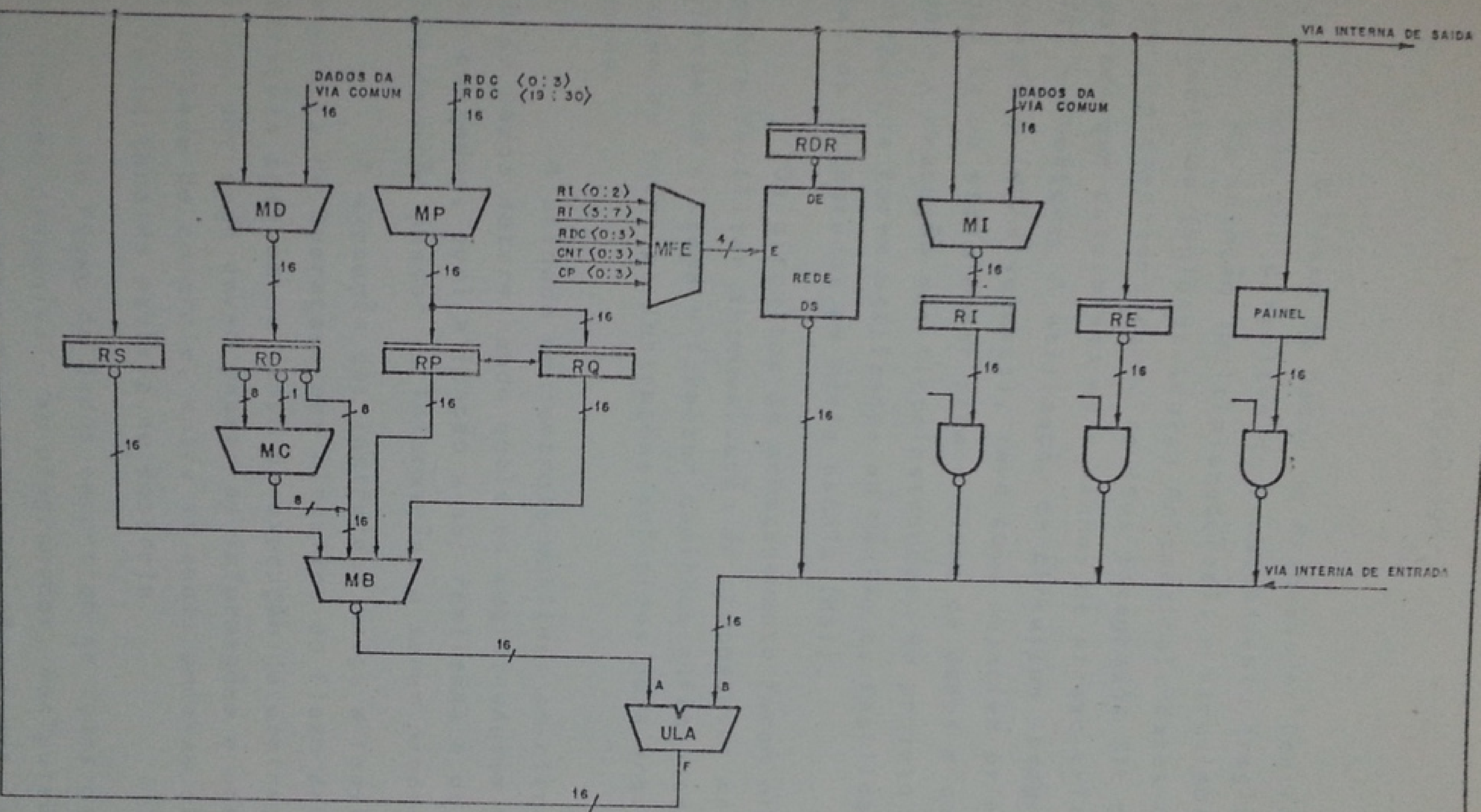


Fig.IV.2. - Fluxo de dados do Processador

1. Fluxo de Dados

O Fluxo de Dados do Processador Central é uma unidade formada por circuitos de armazenamento (registradores) circuitos de seleção (multiplexadores), e circuitos lógicos e aritméticos (Unidade Lógica Aritmética). Estes circuitos foram implementados com circuitos integrados TTL normal e nas partes que determinam os caminhos de atrazo crítico foi usado TTL - Shottcky. A utilização de circuitos integrados TTL - Shottcky (mais velozes), teve como objetivo principal a obtenção de um ciclo menor para o fluxo de dados e consequentemente para a execução das microinstruções. No projeto lógico destes circuitos foram utilizadas ao máximo as facilidades dos circuitos integrados em média escala (MSI).

Os circuitos de armazenamento foram organizados de modo a facilitar a comunicação do processador central com os blocos do sistema e fornecer caminhos suficientes para a execução de todas as instruções definidas pela arquitetura da máquina.

A unidade de controle monitora os circuitos do Fluxo de Dados determinando quais os registradores que deverão ser operados, qual a função a ser realizada e os caminhos por onde deverão passar as informações a serem processadas.

A execução das instruções exige a realização de uma sequência de operações nos circuitos do Fluxo de Dados. Estas operações possuem um tempo de execução determinado pelos caminhos por onde devem passar as informações e por restrições na unidade de controle, entre as quais pode-se citar, o tempo de leitura da memória de controle.

No Fluxo de Dados encontram-se todos os registradores que são disponíveis ao programador, manipulados pelas instruções da máquina e mais os registradores que são manipulados somente pelo controle. Será descrito o Fluxo de Dados da Fig. IV.2., especificando as funções de todos os seus circuitos:

REGISTRADOR DE DADOS (RD)

É um registrador de 16 bits composto por bí- está veis do tipo D, sensíveis à borda de subida do sinal de reló gio (Clock).

A entrada do registrador D é alimentada pela saída do multiplexador D, que seleciona entre a saída da Unidade Aritmética e o caminho de dados da VIA COMUM, qual deve ser copiado no registrador D.

Pode-se ver então, que a função do registrador D é receber e enviar as informações para os outros blocos do sis tema através da Via Comum. Informações vindas da memória pri- mária deverão chegar a ele assim como informações a serem en viadas para a memória primária deverão ser nele carregadas. Portanto a saída do registrador D alimenta o caminho bidire cional de dados da VIA COMUM.

O conteúdo do registrador D pode ser colocado na entrada B da unidade lógica e aritmética. Quando se especifi ca o registrador D para a entrada U.L.A. pode-se colocar seus dois bytes ou apenas o byte menos significativo, colocando em todos os bits do byte mais significativo o bit 7 do pró prio registrador D. Para realizar este tipo de operação exis te o multiplexador C. Esta transformação é útil para execução de instruções que manipulam informações em um só byte. O re gistrador D não é manipulável por instruções e, portanto, o programador não tem acesso a ele, somente o controle (micro programas) pode usá-lo.

Para a implementação do multiplexador D foram esco lhidos 4 módulos 74158, para o registrador D forma escolhidos 4 módulos 74175 e para o multiplexador C foram escolhidos tam bém 2 módulos 74158.

REGISTRADOR P

É um registrador de 16 bits com capacidade de carga e leitura paralela e de deslocamentos à direita ou à esquerda.

A entrada de P é selecionada pelo multiplexador P, podendo nele carregar ou a saída da U.L.A., ou a saída do registrador de dados da memória de controle (RDC) Bits:

0 - 3 , 19 - 30 .

A ligação entre RDC e RP é feita para carregar no Fluxo de Dados as constantes definidas no controle.

A saída do RP pode ser colocada na entrada B da ULA quando selecionada pelo multiplexador B.

O registrador P tem a capacidade de ser deslocado para a esquerda ou para a direita. Quando se realiza um deslocamento em P o bit que sai vai para o "Flip-Flop" de Extensão (E). Para determinar o bit que deve ser inserido no P devido a operação de deslocamento, temos os multiplexadores DD e DE, sendo possíveis Deslocamentos Lógicos, Aritméticos, Giros Simples e Giros intercalando o bit de Extensão. Ainda é possível realizar-se deslocamentos duplos (32 bits) e para isso associa-se o registrador Q ao P, sendo que Q possui os 16 bits menos significativos e P os 16 bits mais significativos. Nos deslocamentos duplos tem-se também os mesmos 4 tipos de deslocamentos simples.

Quando se estiver fazendo um deslocamento aritmético à esquerda o bit de Transbordamento (T) é atualizado com o resultado do "ou exclusivo" dos bits P_{14} e P_{15}, tanto no caso de deslocamentos simples como duplos.

O registrador P não é disponível ao programdor sendo utilizado pelo controle como registrador de trabalho com a principal finalidade de realizar as operações de deslocamentos no fluxo de dados.

Para a implementação do registrador P e Q foram escolhidos 8 módulos 74194, para o multiplexador P foram escolhidos 4 módulos 74158, para os multiplexadores DD e DE foram escolhidos 1 módulo 9313 e 9309 respectivamente.

REGISTRADOR Q (RQ)

É um registrador de 16 bits com as mesmas características do registrador P.

A entrada do registrador Q é alimentada pela saída do multiplexador P, sendo possível carregar em Q somente a saída F da U.L.A.

A saída do Q pode ser colocada na entrada B da ULA através de uma seleção feita pelo multiplexador B.

O registrador Q pode ser deslocado em conjunto com o registrador P, como foi explicado anteriormente, mas não pode ser deslocado separadamente. Portanto a sua principal finalidade é a de prover uma maneira simples e eficiente de fazer deslocamentos e giros duplos no Fluxo de Dados. Ele é utilizado também como registrador de trabalho pelo controle.

O programador não tem acesso ao registrador Q pois as instruções de máquina não fazem referência a ele.

Rede de Registradores

A Rede de Registradores é constituída por uma memória de semicondutor, bipolar, de alta velocidade, de 16 posições de 16 bits cada, formando um conjunto equivalente a 16 registradores de 16 bits.

Essa memória possui uma entrada de endereço de 4 bits, uma entrada de controle de 2 bits, uma entrada de dados de 16 bits e uma saída de dados de 16 bits.

A fonte de endereço para a rede é selecionada pelo multiplexador FE que recebe informação do registrador de instrução (RI), do registrador de dados da memória de controle (RDC), do contador de controle (CNT) e das chaves do painel e coloca-a na entrada de endereço da rede.

Através do registrador de instrução pode-se fornecer endereço para rede usando os bits 0:2 ou 5:7 , pelo registrador DC fornece-se os bits DC 0:3 , pelo contador de controle envia-se os bits CNT 0:3 , e pelo painel usam-se quatro chaves especiais. A fonte de endereço a ser utilizada em um determinado ciclo da unidade central é determinada pelo controle (mais detalhes na descrição do controle).

A entrada de dados da rede é alimentada por um registrador de dados da rede(RDR) que armazena a saída da U.L.A. O registrador DR é de 16 bits composto por bi-estáveis do tipo D, sensíveis à borda de subida do relógio. A sua entrada está conectada diretamente à saída da U.L.A. e sua saída está ligada à entrada de dados da rede. O registrador DR tem a finalidade de armazenar o resultado de uma operação feita na U.L.A. quando se deseja gravá-lo em um dos registradores da rede.

A saída da rede é colocada diretamente na entrada A da Unidade Aritmética, numa ligação em coletor aberto com saídas de outras unidades. Para se selecionar a saída da rede na entrada A da U.L.A. deve-se atuar no sinal SP (seleção da pastilha) da rede que coloca a saída coletor aberto ativa na entrada da U.L.A. Pode-se selecionar os dois bytes da rede, ou somente o byte direito. Para isso o sinal SP é dividido em dois: SPE (Seleção de Pastilha Esquerda) e SPD (Seleção de Pastilha Direita). Somente a seleção do byte direito de um registrador da rede permite a leitura do mesmo, colocando zeros no byte esquerdo, ou a escrita no byte direito de um registrador sem alterar o conteúdo do seu byte esquerdo.

Normalmente a rede está preparada para executar um ciclo de leitura. Somente quando se especifica um ciclo de escrita é que se muda o sinal de controle para que ela escreva o conteúdo do registrador DR em um dos seus 16 registradores.

Nem todos os registradores da rede são disponíveis ao programdor. Descreve-se a seguir, cada um dos 16 registradores especificando as suas funções:

- REGISTRADORES DE PROPÓSITO GERAL: - 8 registradores (R0 - R7) podem ser usados como acumuladores ou como registradores de índice pelo programador, através de especificações nas instruções de máquina.

O registrador R0 é usado como contador de instruções (CI), é um registrador que contém sempre o endereço da instrução corrente. O endereço que fica armazenado no CI é um endereço relativo ao registrador base de programa.

- REGISTRADOR BASE DE PROGRAMA: - É um registrador que contém o endereço da posição de memória onde começa a área de programa. O programador só tem acesso a este registrador através de instruções especiais que não são as mesmas que manipulam os registradores de propósito geral.

- REGISTRADOR LIMITE DE PROGRAMA: - É um registrador que armazena o tamanho da área de programa. Este registrador só pode ser referenciado em instruções especiais.

- REGISTRADOR BASE DE DADOS: - É um registrador que contém o endereço da posição de memória onde começa a área de dados. O programador só tem acesso a ele através de instruções especiais.

Os registradores Base de Dados e Base de Programa são utilizados para cálculo de endereço efetivo nas instruções que referenciam a área de dados e a área de programa da memória respectivamente.

Os registradores bases e limites de dados e programas são também utilizados no esquema de proteção de memória.

- REGISTRADOR RASCUNHO: - São 4 os registradores usados pelo controle, como área de trabalho, na execução das instruções de máquina. Estes registradores não são disponíveis ao programador, pois não existem instruções que os manipulem. Somente o controle pode usá-los para armazenamento temporário.

Portanto, podemos desenhar esquematicamente os 16 registradores dentro da rede:

0	RASCUNHO 0	sem acesso por programa
1	" 1	
2	" 2	
3	" 3	
4	LIMITE DE DADOS	acesso através de instruções especiais
5	LIMITE DE PROGRAMAS	
6	BASE DE DADOS	
7	BASE DE PROGRAMAS	
8	CONTADOR DE INSTRUÇÕES	acesso por instruções normais
9	R1	
10	R2	
11	R3	
12	R4	
13	R5	
14	R6	
15	R7	

A rede de registradores foi implementada com 4 módulos 31Ø1A, o multiplexador das fontes de endereço para a rede com 4 módulos 74S251 e para o registrador DR foram escolhidos 4 módulos 74S175.

- REGISTRADOR_DE_INSTRUÇÕES (RI)

É um registrador de 16 bits composto por bi- estáveis do tipo D, sensíveis à borda de subida do sinal de relógio.

O registrador I tem a finalidade de armazenar a instrução que o processador deve executar. A arquitetura definida possui instruções de uma e de duas palavras. Nas instruções de uma palavra o registrador I armazena a primeira palavra, e o registrador D a segunda. Em ambos os casos os bits da instrução armazenados em I são suficientes para o processador descobrir qual a instrução a ser executada e qual a regra a ser usada no cálculo de endereço efetivo, nas que possuem operando.

A entrada do registrador I é alimentada pelo multiplexador I que seleciona entre o caminho de dados da Via Comum e a saída F da U.L.A. qual deve ser copiado no registrador I. Ao se buscar na memória primária a primeira palavra de uma instrução, tanto o multiplexador I como o D selecionam o caminho de dados da Via Comum para as entradas dos registradores I e D respectivamente. Portanto, a primeira palavra da instrução é colocada tanto em I como em D. Se a instrução for de duas palavras, é feita a busca da segunda palavra, mas desta vez o conteúdo do caminho de dados da Via Comum é copiado somente pelo registrador D.

A saída do registrador I é enviada para o controle onde é identificada a instrução a ser executada. São então excitados os circuitos necessários para realizar-se a sequência de operações pré-estabelecidas para a execução da instrução.

A saída do registrador I é também ligada a um conjunto de portas NE que possuem a saída em coletor aberto, ligadas à entrada A da U.L.A. Desta forma pode-se selecionar o registrador I numa das entradas da U.L.A. para operar-se com ele.

Este registrador não é referenciado por instruções de máquina, portanto o programador não tem acesso a ele. O registrador I é manipulado unicamente pelo controle.

O registrador I foi implementado com 4 módulos 74175 e o multiplexador I com 4 módulos 74158.

- REGISTRADOR DE ENDEREÇO (RE)

É um registrador de 16 bits, formado por bi- estáveis do tipo D, sensíveis à borda de subida do sinal de relógio, com a finalidade de alimentar os fios de endereço da Via Comum quando o processador central está utilizando-a.

A entrada do registrador E está ligada à saída da U.L.A. Este caminho é utilizado para carregar o E com o endereço de uma posição de memória ou com informações de comando para os canais de E/S.

A saída do registrador E também está ligado à entrada A da U.L.A. através de um conjunto de portas NE com coletor aberto.

Este registrador não é referenciado por instrução de máquina, somente manipulado pelo controle.

O registrador E foi implementado com 4 módulos 74175.

- REGISTRADOR DE STATUS (RS)

É um registrador de 16 bits formado por bi - estáveis tipo D sensíveis à borda de subida do sinal de relógio.

O registrador S tem a finalidade de armazenar os indicadores de estado do processador central e está dividido em duas partes, chamadas Direita e Esquerda.

Estes indicadores mostram o estado do processador, armazenando tanto informações devidas a operações realizadas no Fluxo de Dados como informações de posicionamento do sistema.

O registrador S pode ser operado como um registrador de 16 bits ou pode ser manipulado bit a bit pelo controle. As informações que o registrador S contém são as seguintes:

- SD0 - TRANSBORDAMENTO (T) - É um bit que indica se houve um resultado errado numa operação realizada na unidade aritmética ou numa operação de deslocamento aritmético à esquerda. No caso de operações na unidade aritmética o bit T é atualizado segundo o resultado de um "ou exclusivo" entre o vai-um do bit 14 e o vai-um do bit 15 da U.L.A. No caso de deslocamento aritmético a esquerda, o bit T é atualizado segundo um "ou exclusivo" entre osbits 15 e 14 do registrador a ser deslocado.

- SD1 - EXTENSÃO (E) - É um bit usado como extensão dos operandos tanto nas operações realizadas na U.L.A. como no deslocador. No caso de operações na U.L.A. ele é atualizado segundo o vai-um da unidade aritmética. Nas operações com o deslocador o bit que sai é colocado na extensão E.

- SD2 - SINAL (S) - É um bit que armazena o sinal do resultado de uma operação realizada na U.L.A. ou no desloca dor. No caso de operações com a unidade aritmética ele é atualizado segundo o bit F 15 da saída da U.L.A.

S = 1 negativo

S = 0 positivo

- SD3 - ZERO (Z) - É um bit que indica se o resulta do da operação realizada na U.L.A. ou no deslocador é igual a zero.

Z = 1 = zero

Z = 0 ≠ zero

- SD4 - PAR/IMPAR (P) - Este bit indica se o resul tado de uma operação realizada na U.L.A. ou no deslocador é par ou impar.

P = 1 impar

P = 0 par

- SD5 - MODO (M) - Este bit indica em qual estado o processador central está operando. Do ponto de vista do pro gramador existem dois estados possíveis para o Processador Cen tral que serão detalhados mais adiante.

M = 1 supervisor

M = 0 usuário

- SD6 - INTERRUPÇÃO (I) - Este bit indica se o Pro cessador Central pode ou não aceitar interrupções externas.

I = 1 inibidas

I = 0 permitidas

- SD7 - CONDIÇÃO (G) - Este bit é utilizado por al gumas instruções para sinalizar o resultado de uma comparação.

Os bits SE0 a SE7 não são usados pelo controle do Processador Central, podendo ser usados pelo programador, em estado supervisor, definindo-os da maneira que convier.

Os bits SD0-SD7 são atualizados automáticamente após a execução de cada instrução de máquina. Para alterar o estado do bit SD6 é necessário executar uma instrução especial. O bit SD5 pode ser alterado por instruções especiais ou quando ocorre uma interrupção no sistema.

Existem instruções capazes de testar todos os bits do registrador S. Estas instruções permitem ao programador saber como o controle atualizou os bits de S. Além disso existem instruções especiais que encaram o registrador S como um todo, permitindo copiar ou carregar todos os seus bits em outra unidade de armazenamento sob controle de programa. Existem também instruções que modificam os bits de status de acordo com uma máscara.

O tratamento do registrador S como um todo é possível, já que a saída da unidade aritmética está ligada à entrada do multiplexador S e a saída do registrador S está conectada ao multiplexador B, sendo possível selecioná-lo para a entrada B da U.L.A. As outras entradas do multiplexador S estão conectadas aos circuitos geradores dos bits indicadores do estado do sistema.

O multiplexador S foi implementado com 2 módulos 74S158, o byte esquerdo do registrador S foi implementado com 2 módulos 74175, o byte direito com 4 módulos 74S74.

Unidade Lógica e Aritmética (ULA)

A unidade lógica e aritmética tem a capacidade de realizar operações com operandos de 16 bits. Ela possui duas entradas (16 bits): entrada A e entrada B. O resultado de uma operação é fornecida em uma saída de 16 bits que é chamada de F.

A U.L.A. é formada por cinco peças básicas: 4 unidades lógicas e aritméticas de 4 bits cada uma e 1 gerador de vai-um antecipado. Cada uma das unidades de 4 bits usa internamente a técnica de "vai-um antecipado" para realizar as suas funções.Com o auxílio do gerador de vai-um pode-se interligar essas quatro unidades usando a mesma técnica.

Sendo assim, obtem-se uma unidade lógica e aritmética de 16 bits que opera segundo a técnica de vai-um antecipado. Este arranjo foi feito visando obter maior rapidez nas operações lógicas e aritméticas.

Para selecionar qual dos registradores do Fluxo de Dados que entrará na entrada B da U.L.A. tem-se o multiplexador B. Já para a entrada A seleciona-se uma das unidades que participam da ligação em coletor aberto.

A saída F da U.L.A. é enviada a todos os registradores do Fluxo de Dados (como foi explicado anteriormente). Se o controle achar necessário pode copiar o resultado da operação em um desses registradores. Esta saída F é também enviada ao multiplexador do registrador de endereço da memória de controle (MEC).

Aproveitou-se o fato de se ter na entrada A da ULA uma ligação em coletor aberto para criar uma VIA INTERNA DE ENTRADA DE DADOS: Nesta VIA serão conectados o painel, e o registrador de interrupção. O registrador de interrupções recebe os vetores de interrupção enviados pelos canais de E/S que estiverem conectados à Via Comum de Dados. A saída F da U.L.A. é utilizada também para enviar informações para o relógio interno. (VIA INTERNA DE SAÍDA DE DADOS).

Através de instruções especiais o programador terá acesso ao registrador de chaves do painel e ao relógio interno. O registrador de interrupção não é acessado pelo programador, é um registrador de trabalho do controle usado na execução do procedimento necessário para o atendimento de uma interrupção.

Para a implementação da U.L.A. foram escolhidos 4 módulos 74S181 e 1 módulo 74S182. Para o multiplexador B foram usados 8 módulos 9309.

Uma vez descrito o fluxo de dados, identificando-se os módulos MSI usados na sua implementação, e com conhecimento dos algorítmos a serem implementados (ver Apêndice), pode-se passar à descrição do controle do processador central.

2. Controle

Tendo conhecimento do grande número de instruções, da complexidade de alguns algorítmos que o processador deverá executar e levando-se em conta o estudo relatado no capítulo II, torna-se evidente que a melhor sistemática de projeto lógico estruturado, para o processador central, é a que utiliza um núcleo de controle microprogramado. Certamente, o número de passos de controle necessários para a execução de todos os algorítmos é superior ao valor que foi estimado para n_A no capítulo II. (512 > 70).

Sendo assim, para o controle do processador central, foi usado um núcleo microprogramado, relativamente simples, onde se procurou definir uma linguagem de controle dirigida para a implementação dos algorítmos do sistema, estabelecendo-se uma estrutura de microinstruções verticais, não muito codificadas com algumas características horizontais. Procurou-se obter uma solução de compromisso entre os extremos totalmente vertical e horizontal. Para a implementação das microinstruções optou-se por uma estrutura paralela, onde a busca e a execução das microinstruções ocorrem simultaneamente.

No caso de desvios condicionais, procurou-se antecipar o mais possível o instante da decisão, a ponto de escolher ainda dentro do ciclo corrente qual a próxima microinstrução a ser executada.

Esta escolha de filosofia, vertical com sabor horizontal pouco codificada e paralela, para o controle do processador visou obter uma linguagem de controle relativamente simples de se usar (vertical), mas eficiente na manipulação dos recursos do "hardware" (sabor horizontal) e uma frequência de execução de microinstruções relativamente alta, objetivando uma maior rapidez na execução dos algorítmos (paralela e pouco codificada).

Uma vez estabelecida a filosofia do núcleo de controle, passou-se à primeira definição da linguagem de controle. Para que isto fosse possível foi necessário uma análise minuciosa do fluxo de dados proposto, visando a identificação dos principais sinais de controle. O resultado desta análise é que será descrito, apresentando-se somente a solução final adotada, omitindo-se os estágios intermediários que existiram durante o projeto.

2.1. Filosofia e definição da linguagem de controle

No fluxo de dados, as unidades funcionais estáticas que realizam operações transformando as informações são os deslocadores (P e Q) e a U.L.A. A U.L.A. necessita de 6 sinais de controle para ser determinada exatamente a função que deve ser realizada:

M : indica se a operação é lógica ou aritmética

SØ, S1, S2, S3: selecionam uma entre 16 diferentes funções.

C_n : é o bit de vai-um inicial

A tabela de funções possíveis de serem realizadas pela unidade lógica e aritmética do processador é a seguinte:

	M = 1	M = 0 FUNÇÕES ARITMÉTICAS	
S_3 S_2 S_1 S_0	Funções Lógicas	CN = 0	CN = 1
0 0 0 0	$F = \overline{A}$	F = A menos 1	F = A
0 0 0 1	$F = \overline{AB}$	F = AB menos 1	F = AB
0 0 1 0	$F = \overline{A} + B$	$F = A\overline{B}$ menos 1	$F = A\overline{B}$
0 0 1 1	F = 1	F = menos 1	F = 0
0 1 0 0	$F = \overline{A+B}$	$F = A$ mais $(A+\overline{B})$	$F = A$ mais$(A+\overline{B})$mais 1
0 1 0 1	$F = \overline{B}$	$F = AB$ mais $(A+\overline{B})$	$F=AB$ mais$(A+\overline{B})$ mais 1
0 1 1 0	$F = \overline{A \oplus B}$	F = A menos B menos 1	F = A menos B
0 1 1 1	$F = A + \overline{B}$	$F = A + \overline{B}$	$F = (A+\overline{B})$ mais 1
1 0 0 0	$F = \overline{A}B$	F = A mais (A+B)	F = A mais (A+B)mais 1
1 0 0 1	$F = A \oplus B$	F = A mais B	F = A mais B mais 1
1 0 1 0	F = B	$F = A\overline{B}$ mais(A+B)	$F = A\overline{B}$ mais(A+B)mais 1
1 0 1 1	F = A + B	F = A + B	F = (A+B) mais 1
1 1 0 0	F = 0	F = A mais A	F = A mais A mais 1
1 1 0 1	$F = A\overline{B}$	F = AB mais A	F = AB mais A mais 1
1 1 1 0	F = AB	$F = A\overline{B}$ mais A	$F = A\overline{B}$ mais A mais 1
1 1 1 1	F = A	F = A	F = A mais 1

Fig.IV.3-Tabela de Funções da U.L.A.

Já para controlar os deslocadores P e Q são necessários 2 sinais que determinam a operação a ser feita e mais 3 sinais que posicionam os multiplexadores DD e DE especificando qual o tipo de deslocamento a ser executado:

SØ,S1: Selecionam uma entre as quatro seguintes operações nos deslocadores:

00 - Nenhuma operação
10 - Deslocamento à direita
01 - Deslocamento à esquerda
11 - Carga paralela

DESLØ, DESL1: Selecionam um entre os quatro tipos de deslocamentos possíveis.

00 - Deslocamento simples
01 - Deslocamento Aritmético
10 - Giro Simples
11 - Giro com Extensão

DESL2: Determina se o deslocamento é simples (somente P) ou duplo (P e Q).

Para selecionar os dois operandos que deverão participar de uma operação na U.L.A., deve-se posicionar o multiplexador B(entrada B da ULA)em uma de suas quatro alternativas: P, Q, D ou S; e identificar qual unidade deve ser ativada na ligação em coletor aberto da entrada A da U.L.A; as alternativas para esta seleção são:

Nenhuma unidade, REde de registradores, RI e via interna de entrada.

Para armazenar o resultado da operação que for realizada na U.L.A., tem-se as seguintes alternativas: nenhuma unidade,RS, RD, RP, RQ, REDE (dois Bytes), REDE (somente byte direito), RI, RE, via interna de saída, contador de controle.

Se for selecionado para a entrada A da U.L.A.,a rede de registradores ou se se desejar armazenar a saída da ULA na rede, é necessário também especificar qual a fonte de endereço. Isto é feito posicionando o multiplexador das fontes de endereço para a rede, segundo as seguintes alternativas:

- Endereço fornecido pelo registrador I: Este modo de endereçamento é conveniente para operar com os registradores da rede que são manipulados pelas instruções de máquina. Observando o formato das instruções pode-se notar que estes registradores são sempre referenciados através dos bits I 0:2 e I 5:7 . Portanto temos duas possibilidades diferentes para endereçar a rede através do RI.

- Endereço fornecido pelo próprio controle: são necessários quatro bits para permitir ao controle acessar de maneira direta todos os dezesseis registradores da rede.

- Endereço fornecido pelo contador de controle: Os quatro bits do contador de controle endereçam a rede, facilitando assim o acesso sequencial de vários registradores.

- Endereço fornecido pelo painel: Existem quatro chaves especiais no painel que servem para endereçar qualquer um dos 16 registradores da rede.

Com os sinais de controle identificados até agora, já pode-se tirar algumas conclusões. É necessário emitir onze sinais para comandar uma operação no fluxo de dados (U.L.A. e Deslocador). Como o número de funções realizadas pela U.L.A. é muito grande e várias delas não são necessárias para a implementação dos algorítmos, escolhendo-se um subconjunto dessas funções pode-se reduzir o número de sinais de controle agrupando-se num mesmo campo da microinstrução as funções lógicas, aritméticas e os tipos de deslocamentos. Com um campo de 5 bits, trancodificando-o através de uma memória de conteúdo fixo, e expandindo-o para 8 bits, pode-se controlar simultaneamente operações na ULA e no deslocador.

Desta forma consegue-se emitir 32 funções através desse campo de 5 bits, que chamar-se-á FUNC; serão possíveis funções aritméticas ou lógicas ou deslocamentos, ou funções mistas aritméticas ou lógicas e deslocamentos.

Sendo assim para controlar totalmente as operações na U.L.A. e no deslocador serão necessários 8 bits, assim distribuídos em tres campos da microinstrução:

VUQ - C_n : vai um inicial (1 bit)

D - operação do deslocador: Nenhuma Operação
Deslocamento Direita
Deslocamento Esquerda
Carga Paralela (2 bits)

FUNC - Função da U.L.A. e/ou Deslocador (5 bits)

Para selecionar o operando B da U.L.A. são necessários 3 bits divididos em dois campos. Um dos campos, com dois bits, posicionará o multiplexador B e o outro com um bit, posicionará o multiplexador C. O campo C só terá significado se no campo B estiver codificado o registrador D, pois só assim terá sentido a expansão do bit 7 do registrador D através de seu byte esquerdo, como foi explicado na descrição do fluxo de dados.

Para selecionar o operando A da U.L.A. são necessários 8 bits divididos em 3 campos. Um deles, com 2 bits, chamado A, determina qual unidade deve ser ativada na ligação em coletor aberto da entrada A da U.L.A. As alternativas possíveis para esta seleção são as seguintes:

00 - Nenhuma unidade (NOP)
01 - Rede de REgistradores
10 - RI
11 - Via Interna de Entrada

O segundo também com 2 bits, determina qual a fonte de endereço a ser utilizada, se a rede for codificada no campo A. Com qualquer outro código em A este campo não terá significado. O terceiro, com 4 bits, chamado END, determina qual dos 16 registradores da rede será manipulado, se em A for selecionada a rede e em FE estiver especificado endereçamento direto através do controle. Este campo também seleciona qual das unidades pertencentes à via interna de entrada (Reg. de chaves do Painel, Reg. de Endereço (E), Reg. de interrupção) deve ser ativado se em A estiver codificada via interna de entrada.

Para selecionar a unidade que armazenará o resultado da operação realizada são necessários 4 bits, colocados num único campo chamado ARMA. Se neste campo for selecionada a rede de registradores, os campo FE e END realizarão funções semelhantes àquelas, devido à ativação da rede pelo campo A. Já a via interna de saída, ocupará dois códigos neste campo: um para carregar o registrador de dados e outro para o registrador de função do relógio de tempo real do processador.

Como resultado desta análise parcial tem-se a identificação de alguns sinais de controle, podendo-se resumir esquematicamente as conclusões tiradas, definindo-se já alguns campos que deverão existir nas microinstruções:

A	C	B	V	FUNC	FE	D	ARMA	END

Campos responsáveis pela seleção do operando A: A, FE, END
Campos responsáveis pela seleção do operando B: B, C
Campos responsáveis pela operação: FUNC, V, D
Campos responsáveis pelo armazenamento do resultado da operação: ARMA, FE, END

É conveniente ressaltar que, embora os campos B, FE e D sejam codificados, a sua decodificação já se encontra dentro dos módulos MSI que implementam as funções a eles associadas, não sendo necessário nenhum "hardware" especial para a decodificação dos mesmos. O mesmo acontece com as 8 saídas da memória de conteúdo fixo que transcodifica e expande o campo FUNC: A U.L.A. possui no próprio módulo MSI toda a decodificação necessária para os sinais M,S0, S1,S2 e S3; os multiplexadores DD e DE também fazem o mesmo com os sinais DESLØ, DESL1 e DESL2. O campo END, quando usado para endereçar a rede de registradores, também é decodificado no próprio módulo MSI da rede. Quando ele é usado para a via interna de entrada não é feita nenhuma decodificação.

Desta forma, dos campos já definidos somente os campos A e ARMA é que possuem decodificação direta (um nível só).

Com os campos já definidos o controle é capaz de comandar todo o fluxo de dados, sendo possível, inclusive, operar simultaneamente a U.L.A. e o deslocador.

Para que o controle seja capaz de comandar todo o sistema é necessário criar mais um campo onde seriam codificados microordens do tipo: Ler e escrever na memória primária, requisitar um ciclo da via comum, desviar da sequência de controle (pulos), reconhecer interrupção, liberar o sistema de interrupção, ler vetor de interrupção do canal e em geral outras operações especiais.

Associado às operações especiais que envolvem a realização de um diálogo de utilização através da via comum, o processador deve ter condições de saber se este diálogo já acabou. Isto é conveniente, pois nos casos em que é realizado um diálogo longo, por exemplo, uma leitura de uma posição da memória primária, se o processador tiver condições de realizar outras operações, independentes do dado a ser lido,enquanto a memória estiver fazendo o acesso, o processamento não precisa parar. Somente no instante em que o processador precisar operar sobre o dado lido, ele deve testar primeiramente, se o diálogo de utilização já foi finalizado, ou seja,se odado está no registrador D.

Com esta característica, torna-se possível a operação simultânea do processador central e da memória primária. O mesmo acontece entre o processador e os canais de entrada e saída de dados.

Para a realização de desvios condicionais, testando os mais diversos indicadores de estado necessários ao sequencializador, é preciso que se identifique qual desses indicadores deve ser usado, durante a realização de uma operação especial de desvio condicional.

Desta forma, pode-se identificar mais três campos que deverão pertencer às microinstruções. Um campo para as operações especiais, que será chamado de ESP, possuindo 5 bits, sendo possível a codificação de até 32 diferentes operações. Um outro campo para controlar o sincronismo entre o sequencializador do controle do processador e o controle da via comum de comunicação (C.V.C.C.). Este campo será chamado S e possuirá um único bit. A sua função será a de determinar uma parada na sequência de execução das microinstruções, se anteriormente foi feita uma requisição para utilizar a via comum e o diálogo decorrente desta utilização ainda não terminou;se o diálogo já terminou ou se anteriormente não foi feita uma requisição da via comum, ele simplesmente não causa a parada, prosseguindo o processamento das microinstruções na sequência normal.

O terceiro campo, é aquele que identifica qual indicador deve ser testado, devido à execução de uma operação especial de desvio condicional. Este campo será chamado de COND e constará de três bits. Com 3 bits tem-se somente 8 condições para teste, que é um número pequeno. Sendo assim, pode-se incorporar o bit do campo S aos bits do campo COND, quando aparecer no campo ESP uma microordem condicional. Desta forma tem-se 16 condições de teste, com uma única restrição: o campo S perde o seu significado quando estiver sendo usado para identificar uma variável de teste.

O campo ESP possui codificação direta de um nível. O campo COND possui também uma decodificação direta de um nível só que esta decodificação já está incorporada nos módulos

escolhidos para implementar o multiplexador de condições do sequencializador (controle).

Com estes 3 novos campos, tem-se praticamente definido o formato das microinstruções, uma vez que, um dos objetivos desta definição é a obtenção de uma linguagem de controle relativamente simples de se microporgramar, não será definido microinstruções com diferentes formatos ou mesmo ocupando mais de uma palavra da memória de controle.

Como resultado da análise feita definiu-se microinstruções com 32 bits e cujo formato é o seguinte:

S	A	C	B	V	FUNC	FE	D	ARMA	ESP	COND	END

2.2. Implementação do núcleo de controle

Estabelecida a filosofia e o formato das microinstruções que o controle deverá interpretar, pode-se passar à definição do núcleo de controle.

Para que esta definição seja feita eficientemente, deve-se novamente analisar os algorítmos a serem implementados e procurar identificar algumas características especiais que deverão existir no núcleo de controle, com a finalidade de tornar mais fácil ou mais eficiente a implementação desses algorítmos.

Procurou-se definir um núcleo de controle que fosse bastante simples na sua concepção, mas que implementasse eficientemente os algorítmos desejados.

Fig. IV.4. - Núcleo de controle do Processador

O núcleo escolhido assemelha-se bastante com o núcleo fundamental apresentado no capítulo II; simplesmente foram adicionadas uma unidade contadora, cuja finalidade seria a de facilitar as operações repetitivas, e duas unidades armazenadas, cuja finalidade seria a de armazenar o endereço de retôrno de uma micro subrotina, tornando possível a implementação de microsubrotinas em 2 níveis.

Da própria definição do formato das microinstruções, ficou estabelecido que tanto a memória de controle como o seu registrador de dados teriam 32 bits. Da análise dos algorítmos também concluiu-se que a maioria dos microprogramas deveria compreender a execução de um conjunto de microinstruções colocadas em posições consecutivas. Portanto, pré-estabeleceu-se, com exceção das micro instruções de desvio, que a próxima microinstrução a ser executada seria aquela que estivesse no endereço seguinte à microinstrução corrente. Para facilitar a operação que soma uma unidade ao conteúdo do registrador de endereço da memória de controle, foi colocada uma unidade somadora de 1. Optou-se por esta solução à um contador para o registrador de endereço da memória de controle, para se obter o endereço seguinte mais rapidamente. Isto é bastante importante se for lembrado que, pela filosofia definida, o controle executará microinstrução de uma maneira paralela, e no caso de desvios, a decisão e o acesso da próxima microinstrução devem ser feitos no mesmo ciclo.

A seguir apresenta-se o esquema do núcleo de controle definido e depois passa-se à descrição de cada uma das suas unidades: (Figura IV.4.)

Memória de Controle (MC): É a memória onde devem ser armazenados os microprogramas que implementam os algorítmos das instruções do processador. Esta memória está organizada em palavras de 32 bits e pode ser implementada através de uma memória de conteúdo fixo (ROM) ou de uma memória escritível. Na presente implementação do processador foi usado uma MC escritível contituída por módulos MSI- TTL bipolar bastante velozes (3101A), conseguindo-se assim um tempo de leitura da ordem de 80ns.

Registrador de Dados da Memória de Controle (RDC)

É um registrador de 32 bits, constituídos por bi-estáveis do tipo D sensíveis à borda de subida do sinal de tempo. A função do registrador DC é armazenar a microinstrução que deve ser executada durante um ciclo do controle. O registrador DC foi implementado com 8 módulos MSI 74S175.

Registrador de Endereço da Memória de Controle (REC)

É um registrador de 12 bits constituído por bi-estáveis do tipo D, sensíveis à borda de subida do sinal de tempo. O registrador EC fornecerá o endereço da microinstrução que deve ser lida na memória de controle. A sua entrada está conectada à saída do multiplexador EC. A saída deste registrador também é enviada ao multiplexador RØ, com a finalidade de armazenar o endereço de retorno no caso de um pulo para uma microsubrotina, e também para a unidade somadora de 1, cuja finalidade é gerar o endereço da próxima microinstrução. O controle de carga do registrador EC é feito por um sinal que vem da lógica de endereçamento do controle.

O registrador EC possuindo 12 bits proporciona um espaço de endereçamento para a memória de controle de 4K palavras.

O REC foi implementado com 6 módulos 74S74.

Registradores de retorno (RR): Existem dois registradores de retorno, cada um com 12 bits. Eles são constituídos por bi-estáveis do tipo D sensíveis à borda de subida do sinal de tempo. Estes dois registradores formam uma pilha "last-in","first-out", onde á armazenado o conteúdo do REC no caso da execução de uma microinstrução que determina um pulo para uma microsubrotina.

No caso da execução de um retorno de uma microsubrotina, o conteúdo do registrador topo da pilha (RRØ) é carregado no REC e o RRØ é carregado como o conteúdo do RR1 e este por sua vez é feito igual a zero.

Para que estes dois registradores operem como uma pilha de endereços, eles necessitam de dois multiplexadores, MRØ e MR1, que alimentam as entradas de RRØ e RR1 respectivamente, e de uma unidade que sinaliza se a pilha está vazia ou cheia. RRØ e RR1 foram implementados com 6 módulos 74175 e MRØ e MR1 com 6 módulos 74158. O contador CNS foi implementado com 1 módulo 74193.

Contador de controle (CNT): É constituído por uma unidade contadora de 4 bits, cuja principal finalidade é proporcionar aos microprogramas a realização de operações repetitivas, de maneira simples. Existe uma variável de teste que sinaliza a presença do CNT no estado $(0000)_2$. A entrada deste contador é alimentado pelo multiplexador CNT que seleciona entre os bits RDC 0:3 ou F 0:3 (saída da U.L.A.) qual que deve ser copiado no CNT, sob comando de uma microinstrução especial. O contador CNT foi implementado com um módulo 74193 e o MCNT com um módulo 8266.

Multiplexador do registrador de endereço da memória de controle (MEC): O MEC é responsável pela seleção da unidade que deve fornecer o endereço para o REC, no decorrer da execução de uma microinstrução. O MEC é constituido por unidades multiplexadoras que selecionamuma entre oito diferentes entradas. Para isso ele possui 3 sinais responsáveis pela seleção que são fornecidos pelo circuito da lógica de endereçamento (LE).

O circuito que implementa a lógica de endereçamento, recebe informações do RDC (campo ESP), do multiplexador de condições (sinal verdade ou falso) e do circuito base de tempo. Com estas informações ele é capaz de gerar os sinais de seleção do MEC e os sinais de tempo (clock) para todos os registradores do núcleo de controle. Este circuito possui como peça principal na sua implementação uma memória de conteúdo fixo.

As oito possíveis entradas para o MEC são as seguintes:

<u>. + 1</u> : Saída da unidade somadora de um, cuja finalidade e gerar o endereço da próxima microinstrução numa sequência normal.

<u>.2 entradas vindas do circuito identificador:</u> São duas entradas que fornecem um endereço associado ao conteúdo do registrador de instrução do processador (RI).

O circuito identificador (mapeador) simplesmente mapeia o campo de operação das instruções de máquina no espaço de endereçamento da memória de controle, identificando as microrotinas responsáveis pela execução da instrução que se encontra no registrador I.

Uma dessas entradas é responsável pelos endereços ou de microrotinas de cálculo de endereço ou de execução. Para determinar a cópia do conteúdo desta entrada no REC existe uma microinstrução especial no campo ESP. A outra entrada é responsável por endereços de microrotinas de preparação de dados ou de cálculo de endereço, possuindo também uma outra microinstrução no campo ESP que ordena a sua cópia no REC.

Estas duas microinstruções podem ser encaradas genericamente, como sendo um desvio condicional ao conteúdo do registrador I, sendo que o campo de endereço é fornecido pelo

circuito identificador (mapeador). Elas são responsáveis pela troca de informações entre o circuito identificador e o sequencializador.

Na implementação do circuito mapeador novamente foram usadas memórias de conteúdo fixo.

<u>. F 0:11</u>: Esta entrada é alimentada pelos 12 bits menos significativos da U.L.A., e tem como finalidade principal o endereçamento de uma microrotina através do conteúdo de um dos registradores do fluxo de dados. Para comandar a cópia desta entrada no REC existe uma outra microinstrução no campo ESP.

<u>.Código de Interrupção:</u> Esta entrada é alimentada pelo circuito responsável pelo mapeamento das interrupções. Quando o processador recebe um pedido de interrupção é gerado um código associado ao pedido de maior prioridade naquele instante; este código endereça uma memória de conteúdo fixo, que conterá o endereço da microrotina de interrupção necessária para o seu tratamento. O conteúdo da memória de interrupção só é copiado no REC sob comando de uma microinstrução especial. Esta microinstrução ordena a cópia somente se o contador de microsubrotina estiver no estado $(00)_2$. Esta microinstrução está codificada no campo ESP e ela determina o fim da execução de uma instrução de máquina ou o retorno de uma microsubrotina, dependendo do estado do CNS.

<u>.Chaves do painel:</u> Esta entrada é alimentada pela saída de uma memória de conteúdo fixo que mapeia os diferentes comandos via painel no espaço endereçável da memória de controle. As chaves do painel que possuem microrotinas associadas a elas, alimentam as entradas de endereço da memória do painel. Como saída a memória fornece os endereços das microtinas.

Para copiar estes endereços no REC existe também uma microinstrução no campo ESP. Para o microprograma saber o exato instante para emitir esta microinstrução existe uma variável de teste. Esta variável determina se uma das chaves que possuem microrotinas associadas a elas, foi acionada.

. Endereço de retorno (RRO): Esta entrada é alimentada pelo registrador RRØ e a sua finalidade é fornecer o endereço de retorno de uma microsubrotina. Este endereço é copiado no REC sob comando de uma microinstrução especial codificado no campo ESP. Esta microinstrução é a mesma que ordena a cópia do código de interrupção quando o CNS estiver no estado $(00)_2$ e existir um pedido de interrupção; caso não exista pedido de interrupção, é copiado o conteúdo de RRO. Quando o CNS estiver no estado $(00)_2$ o conteúdo de RRO deve ser $(0.....0)_2$ que é o endereço da microrotina de busca de instruções.

.DC 19:30: Esta entrada é alimentada pelo registrador de dados da memória de controle e a sua finalidade é fornecer um endereço quando é executada uma microinstrução de desvio da sequência normal. No campo ESP existem microinstrução de desvio, desvio para microsubrotina, podendo ser condicionais ou incondicionais.

Multiplexador de condições: Esta unidade é responsável pelo resultado dos testes feitos através de microinstruções com as variáveis de teste do sistema. O multiplexador de condições recebe informações do RDC (campos COND,S) para poder eleger qual das 16 condições de entrada devem ser testadas no presente ciclo. Como resultado do teste é fornecido um sinal que indica se a condição é verdadeira ou falsa. Este sinal é utilizado pela lógica de endereçamento para coordenar os endereços da memória de controle na sequência correta. Todas as variáveis de teste foram geradas com o menor número de atrasos lógicos possíveis, para permitir a decisão do próximo endereço ainda dentro do ciclo corrente.

O multiplexador de condições foi implementado com dois módulos 74S251.

<u>Decodificador e Transcodificador:</u> Através das saídas do RDC é que serão gerados os sinais de controle, correspondentes a cada microordem num campo da microinstrução, que irão controlar as unidades funcionais e as vias do fluxo de dados do processador. Alguns campos da microinstrução são transmitidos diretamente para o fluxo de dados (C,B,V,FE,D,END), outros são decodificados, gerando sinais de controle mutuamente exclusivo (A, ESP, ARMA) e outros são transcodificados (FUNC).

As unidades transcodificadoras foram implementadas com módulos 82S123 e as unidades decodificadoras foram implementadas com módulos 3205.

<u>Tempo e Sincronismo</u>: Este circuito é responsável pela geração de todos os sinais de tempo que irão comandar as mudanças de estado e as trocas de informação no sistema e pelo esquema de sincronismo quando o processador utiliza a via comum.

O circuito de sincronismo recebe sinais de comando do RDC e do controlador da via comum para decidir se o processador precisa esperar para receber uma informação pedida ou se ele já pode prosseguir pois a informação já se encontra no registrador D. Caso o circuito de sincronismo decida que o processador precisa esperar, ele bloqueia a geração dos sinais de tempo, congelando todas as operações até que chegue o sinal FIM da via comum terminando o ciclo de utilização. Nesta altura são reativados os sinais de tempo e o processador prossegue, já sabendo que a informação pedida encontra-se em seu poder. O campo S da microinstrução é que controla o circuito de sincronismo.

O circuito de tempo é constituido basicamente por um oscilador na frequência de 25MHZ e de um registrador deslocador em anel ("ring -counter"). Os sinais básicos de tempo gerados são os seguintes: (Fig. IV.5.).

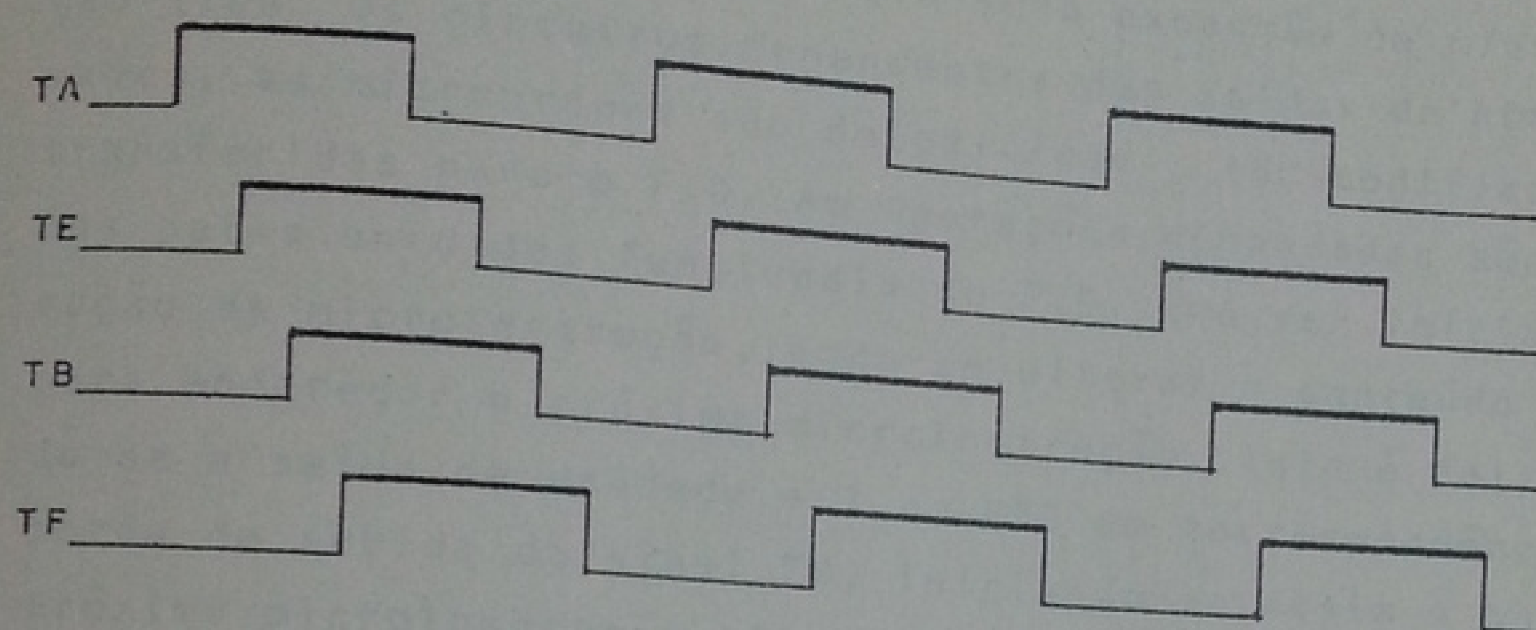


Fig. IV.5. - Sinais básicos de tempo

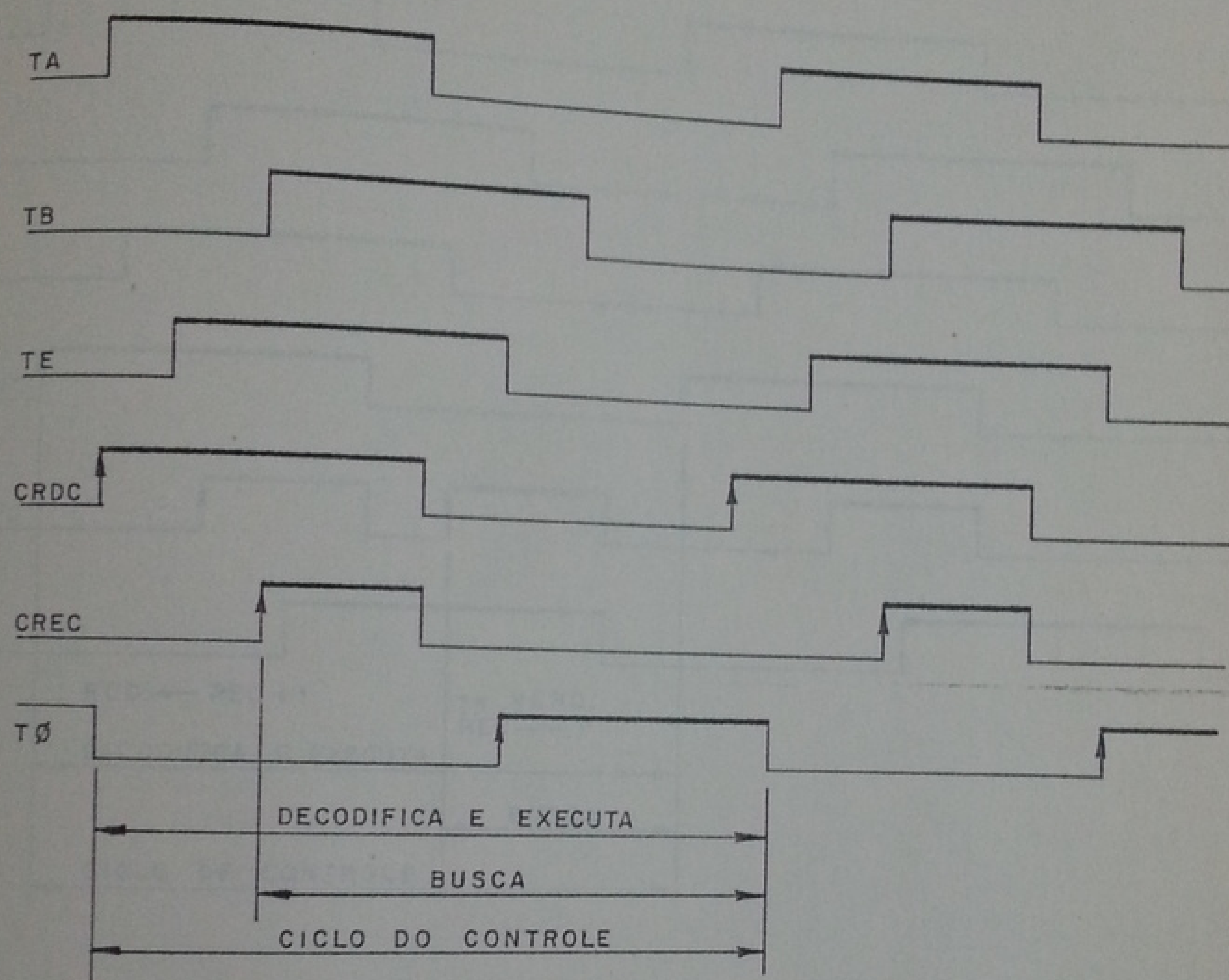
2.3. Diagrama de tempo das microinstruções

Com a finalidade de se obter um ciclo de fluxo de dados de pequena duração, foram escolhidos módulos MSI super rápidos nos caminhos críticos do F.D. e do controle. A operação mais demorada no FD é a leitura de um registrador da rede, operando-o na U.L.A. e depois armazenando-o novamente na rede. O tempo necessário para a execução dessa circulação de informação através do F.D. é que determina o seu ciclo. Levando-se em conta os atrasos máximos associados a cada elemento componente dessa via, calculou-se um tempo da ordem de 120ns para leitura da rede e operação na U.L.A. e mais 50ns para escrita na rede.

Com estes dados e mais o tempo de acesso da memória de controle (da ordem de 80ns) pode-se determinar um diagrama de tempos para a execução das microinstruções.

Na borda de subida do sinal de tempo TA, é copiado o conteúdo da saída da memória de controle no registrador DC. A partir deste instante inicia-se a execução da microinstrução lida. Os circuitos dependentes das saídas de RDC são ativados, as microordens são decodificad , tanscodificadas e transferidas para o F.D. As operações comandadas são iniciadas pelas unidades funcionais do F.D. Uma vez iniciada a execução da microinstrução, pode-se alterar o conteúdo do REC para endereçar a próxima microinstrução. Isto é feito copiando-se a saída da unidade + 1 no REC em correspondência com a borda de subida do sinal TB, iniciando-se assim a busca da próxima microinstrução. A operação iniciada no FD leva no máximo 120ns para ter o seu resultado pronto para ser armazenado em um registrador. Desta forma na borda de descida do sinal TE esse resultado é armazenado no registrador especificado pelo campo ARMA. Novamente em correspondência com o sinal TA é copiado a nova microinstrução no RDC e o ciclo se repete. (Fig.IV.6.).

Se a microinstrução que for copiada no RDC especificar através do campo ESP um desvio condicional da sequência de controle, o sequencializador deve saber o resultado do teste pelo menos 80ns (tempo de acesso da memória de controle) antes do fim do ciclo do controle, para que ele possa decidir corretamente qual o endereço da próxima microinstrução. Os circuitos geradores de condições (variáveis de teste) foram implementados com circuitos super rápidos para permitir que esta decisão fosse tomada o mais cedo possível. Levando-se em conta os atrasos dos elementos lógicos que constituem estes circuitos, conclui-se que após 120ns do início de uma operação no F.D. o sinal de saída do multiplexador de condições está corretamente atualizado e pronto para orientar as decisões. Desta forma, se for o caso, um novo endereço é copiado no REC em correspondência com a borda de descida do sinal TE. Para este caso é válido o seguinte diagrama de tempos: (Fig.IV.7.).



CRDC: cópia no RDC
CREC: cópia no REC
TØ: armazenamento do resultado

Fig. IV.6. - Diagrama de tempo de microinstruções sem desvio da sequência normal

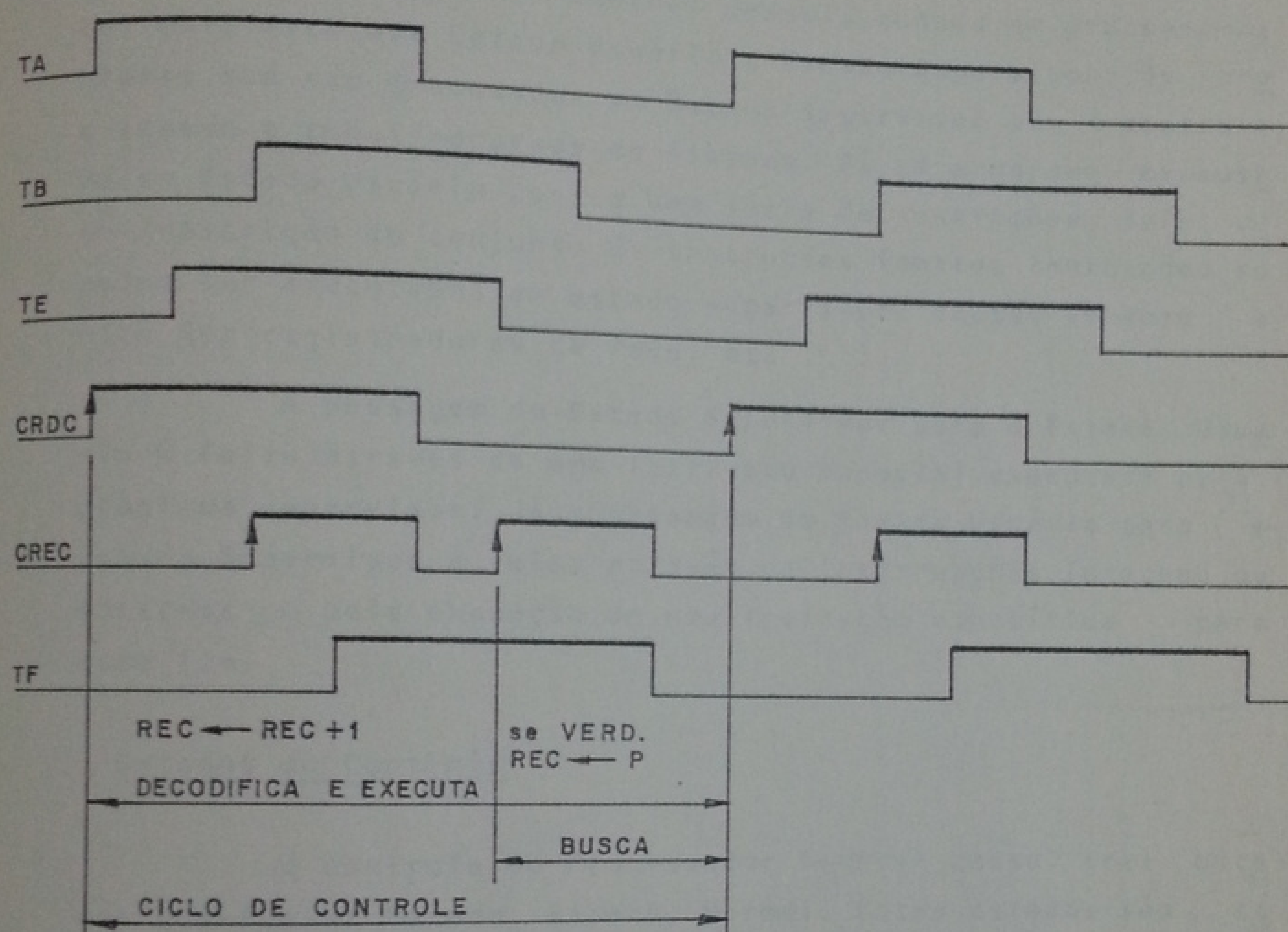


Fig. IV.7. - Diagrama de tempo de microinstruções com desvio condicional e decisão no pró prio ciclo.

2.4. Estados de Processamento e do Controle

- Estados de Processamento

O Processador Central possui, quando em processamento, dois estados: Estado Usuário e Estado Supervisor. Os programas que são executados em Estado Supervisor tem controle e acesso a todos recursos do sistema. Já um programa executado em Estado Usuário possue uma série de restrições, tais como: restrição do conjunto de instruções (certas instruções só podem ser executadas em estado supervisor) acesso somente a oito dos registradores da rede, etc .

A passagem do Estado Supervisor para o Estado Usuário é feita através de uma instrução especial executada pelo programa supervisor. Já a passagem do Estado Usuário para o Estado Supervisor é feita através de interrupções internas ou externas ou pela execução de uma instrução específica para esse fim.

- Estados do Controle

O controle do Processador Central possui tres estados possíveis: Parado, Espera, Normal. Estes estados são caracterizados por procedimentos executados por micro-programas. Estes procedimentos que determinam um estado são caracterizados por uma micro-rotina cíclica (em "loop"). Essas micro-rotinas só saem do ciclo devido a ocorrência de evento externo ao controle. Ver diagrama 1.

Quando liga-se o computador, o controle deve executar uma micro-rotina que testa as chaves do painel. Nesta hora o processador encontra-se no estado parado e o painel determina as operações a serem executadas pelo controle. Se for posicionada a chave que indica carga, o controle sai do ciclo de teste de chaves, executa a micro-rotina de carga via painel e retorna ao teste de chaves permanecendo no estado Parado.

Se a chave pressionada for MOSTRA o controle executa a micro-rotina associada a essa chave e retorna ao ciclo de teste de chaves (Estado Parado). Se for posicionada a chave PARTIDA provoca-se uma mudança de estado. O controle entra no ciclo normal de execução de instruções. Neste ciclo o processador pode receber interrupções. As interrupções enviadas ao processador só são atendidas no fim da execução de uma instrução de máquina e antes da busca da próxima instrução (ver diagrama 2.).

Se durante o ciclo normal for executada uma instrução PARE ou for apertado o botão PARE do painel, o controle volta para o ciclo de teste de chaves (Estado Parado). Se for executada a instrução ESPERE ou for apertado o botão ESPERE do painel, o controle passa para um ciclo de teste de interrupção (Estado de Espera). Assim que chegar um pedido de interrupção o controle passa para o estado normal de execução de instruções.

Pode-se resumir esquematicamente as mudanças de estado do controle do processador central no seguinte esquema:

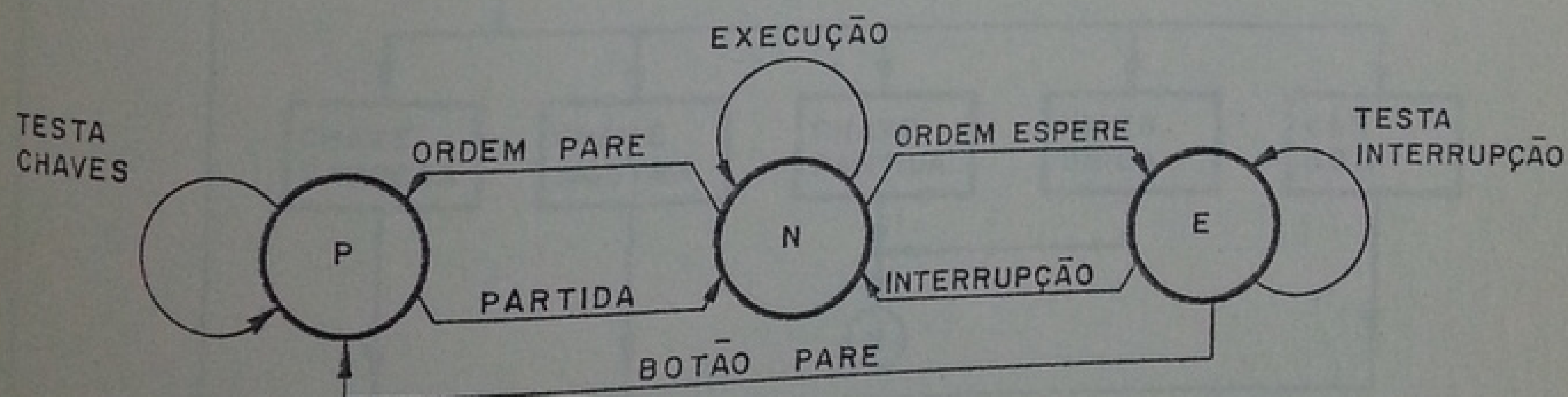


Diagrama 1

Fig. IV.8. - Estados do Controle

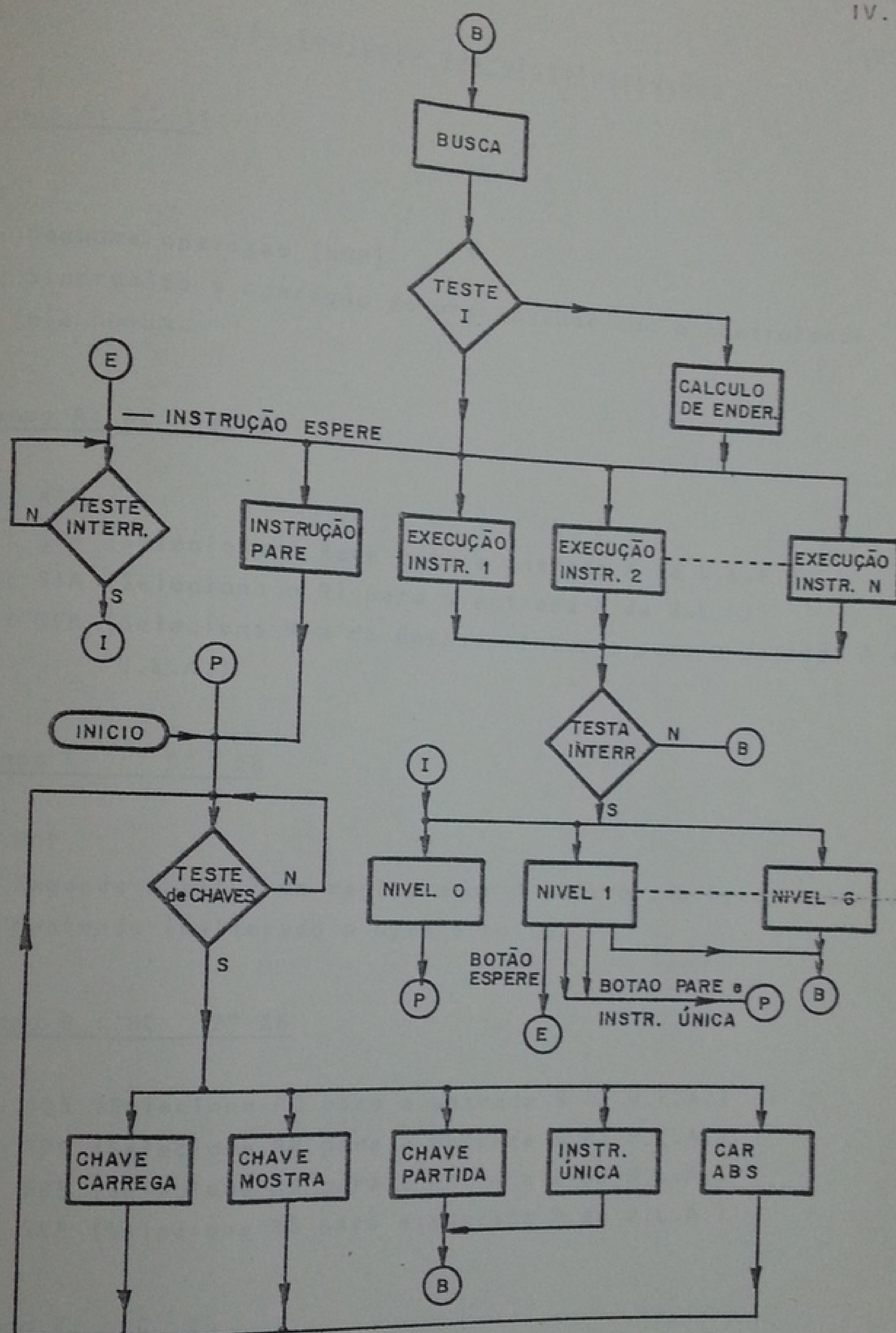


Fig.IV.9. - Diagrama de Execução dos microprogramas no processador central.

Diagrama 2

2.5. Conjunto das microinstruções

Campo S: DC 31

S:

0 : Nenhuma operação (NOP)

1: Sincroniza a operação do processador com o controlador da via comum.

Campo A: DC 30:29

00: NOP

01: SRA (Seleciona a Rede para a entrada A da U.L.A.)

10: SIA (Seleciona o RI para a entrada A da U.L.A.)

11: SVA (Seleciona Via de Entrada Interna para a entrada A da U.L.A.)

Campo C : DC 28

0: NOP

1: Expande o bit 7 do registrador D para o seu byte esquerdo mantendo inalterado o byte direito.

Campo B : DC 27:26

00: SQB (Seleciona RQ para a entrada B da U.L.A.)

01: SPB (Seleciona RP para a entrada B da U.L.A.)

10: SDB (Seleciona RD para e entrada B da U.L.A.)

11: SSB (Seleciona RS para a entrada B da U.L.A.)

Campo V: DC 25

0: operação sem vai-um inicial

1: operação com vai um inicial

Campo FUNC: DC 24:20

00000 -	LS	Lógico simples
00001 -	AS	Aritmético simples
00010 -	GS	Giro simples
00011 -	GES	Giro com E simples
00100 -	LD	Lógico duplo
00101 -	AD	Aritmético duplo
00110 -	GD	Giro duplo
00111 -	GED	Giro com E duplo
01000 -	F=A	sem atualização de status
01001 -	F=B	" " "
01010 -	$F=\overline{B}$	" " "
01011 -	F=A∧B	" " "
01100 -	F=A∨B	" " "
01110 -	F=A⊕B	" " "
01110 -	DES	Lógico simples sem atualizar E
01111 -	DED	Lógico duplo sem atualizar E
10000 -	F=A-1	com atualização de status
10001 -	F=A-B-1	" " "
10010 -	F=A+B	" " "
10011 -	F=A	" " "
10100 -	$F=A+\overline{B}$	" " "
10101 -	F=A∧B	" " "
10110 -	F=A∨B	" " "
10111 -	F=A⊕B	" " "
11000 -	F=A-1	sem " "
11001 -	F=A-B-1	" " "
11010 -	F=A+B	" " "
11011 -	F=A	" " "
11101 -		
11110 -		
11111 -		

Campo FE : DC 19:18

00: EI 0:2 o endereço vem do I bits Ø a 2
01: EI 5:7 o endereço vem do I bits 5 a 7
10: EEND o endereço vem do RDC bits Ø a 3
11: ECNT o endereço vem do CNT bits Ø a 3

Campo D: DC 17:16

00: NOP
01: Deslocamento a esquerda
10: Deslocamento a direita
11: Carga paralela

Campo ARMA: DC 15:12

0000 -	NOP	
0001 -	CNT	armazena no CNT
0010 -	S	armazena no RS
0011 -	D	" no RD
0100 -	P	" no RP
0101 -	Q	" no RQ
0110 -	R	" no REDE
0111 -	I	" no RI
1000 -	E	" no RE
1001 -	RELD	" no Relógio -DADOS
1010 -	RD	" somente no byte direito da Rede
1011 -	RELF	" no Relógico - Função
1100 -	LIVRE	
1101 -	LIVRE	
1110 -	LIVRE	
1111 -	LIVRE	

Campo ESP: DC 11:7

00000 -	NOP	
00001 -	FIM	Fim de execução incondicional
00010 -	FIMC	Fim de execução condicional
00011 -	SEB	Posiciona para Busca
00100 -	PLAC	Pulo condicional
00101 -	Livre	
00110 -	PUGC	Pula e guarda condicional
00111 -	FOR1	Copia entrada 1 do mapeador no REC
01000 -	FOR2	Copia entrada 2 do mapeador no REC
01001 -	PAIN	Copia entrada da memória do Painel no REC
01010 -	FUNC	Copia F 0:11 no REC
01011 -	SALC	Salto condicional
01100 -	DESC	Decrementa CNT ou salta se ZERO
01101 -	SEVI	Seleciona vetor de interrupção
01110 -	REPC	Repete até que CNT=0 ou a condição especificada em COND seja verdadeira.
01111 -	LIVRE	
10000 -	LE	Requisição de um ciclo de leitura na memória
10001 -	RVC	Requisição da via comum
10010 -	LIVRE	
10011 -	LIVRE	
10100 -	CNT	Copia RDC 0:3 em CNT
10101 -	MODS	Altera o bit de S especificado em END segundo a saída Z da U.L.A. ou saída do circuito gerador de condições
10110 -	LIVRE	
10111 -	ESC	Requisição de um ciclo de escrita na memória
11000 -	RECI	Reconhece interrupção
11001 -	LINT	Limpa interrupção
11010 -	PERI	Libera sistema de interrupção
11011 -	EEX	Muda o sentido da entrada do Mapeador
11100 -	SETE	Posiciona byte esquerdo para a memória
11101 -	PAR	Endereça a rede via chaves do painel
11110 -	LIVRE	
11111 -	LIVRE	

Campo COND : DC 31 ; DC 6:4

0000	-	NOP	
0001	-	E	Incondicional
0010	-	T	Testa extensão
0011	-	Z	Testa Transbordamento
0100	-	S	Testa zero na U.L.A.
0101	-	M	Testa sinal da U.L.A.
0110	-	QI	Testa estado supervisor/usuário
0111	-	P	Testa bit 1 do reg.. Q
1000	-	LONG	Testa bit menos significativo da U.L.A.
10001	-	IND	Testa instrução longa
1010	-	INX	Testa indireto de registrador
1011	-	INT	Testa indexado
1100	-	CHA	Testa interrupção
1101	-	ZCNT	Testa chave do painel
1110	-	BP	Testa zero no CNT
1111	-	LIVRE	Testa uso de base de programa

Campos END: DC 3:0

Este campo pode ter vários significados: Se no campo A for especificado C então o campo END possui o seguinte significado:

0001: E   Seleciona E para entrada A de U.L.A.
0010: P   Seleciona chaves do Painel para entrada A de U.L.A.
0100: Livre
1000: VI Seleciona vetor de interrupção para entrada A de U.L.A.

Se no campo ESP for especificado MODS então o campo END possui o seguinte significado:

0000	-	T	Transbordamento
0001	-	E	Extensão
0010	-	S	Sinal da saída da U.L.A.
0011	-	Z	Zero na saída da U.L.A.
0100	-	P	Bit Ø da saída da U.L.A.
0101	-	M	Modo supervisor / usuário
0110	-	I	Permite/Inibe interrupção
0111	-	C	Bit de Condição

Se o campo END for usado como uma constante, seu conteúdo é encarado como um número binário. Isto acontece se no campo ESP for especificado CNT.

Se o campo END for usado para endereçar um registrador da rede, então ele possui o seguinte significado:

0000	-	RASØ	Rascunho Ø
0001	-	RAS1	Rascunho 1
0010	-	RAS2	Rascunho 2
00111	-	RAS3	Rascunho 3
0100	-	LD	Limite de dados
0101	-	LP	Limite de programas
0110	-	BD	Base de dados
0111	-	BP	Base de Programas
1000	-	CI	Contador de Instrução (RØ)
1001	-	R1	Registrador geral 1
1010	-	R2	Registrador geral 2
1011	-	R3	Registrador geral 3
1100	-	R4	Registrador geral 4
1101	-	R5	Registrador geral 5
1110	-	R6	Registrador geral 6
1111	-	R7	Registrador geral 7

Convém ressaltar, antes de encerrar esta descrição que todas as soluções apresentadas nos capítulos III e IV foram implementadas eletricamente, existindo um protótipo deste minicomputador no Laboratório de Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

V. APÊNDICE

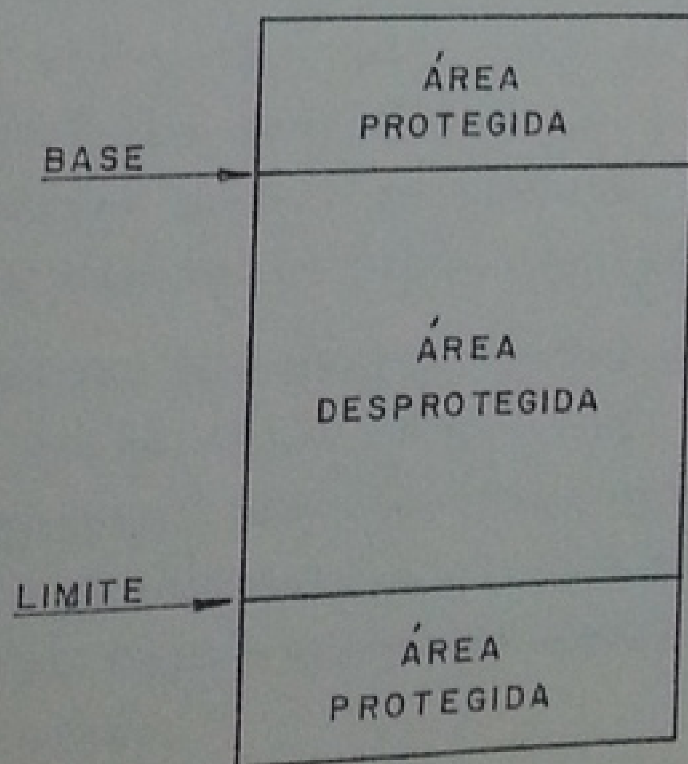
Descrição resumida da arquitetura do processador

1. Filosofia de Proteção de Memória

O espaço de armazenamento primário do processador possui um esquema de proteção baseado na noção de ponteiro base e limite.

Estes ponteiro indicam a área de armazenamento disponível a um determinado processo. Será usado o termo processo para designar um programa em execução. A área fora da marcação dos ponteiros é protegida, não sendo permitido acesso a essa área, nem para ler nem para escrever.

O ponteiro base conterá o endereço inicial da área de armazenamento desprotegida e o ponteiro limite conterá o tamanho dessa área, esquematicamente tem-se:



Qualquer tentativa de acesso a uma posição de memória numa área de armazenamento protegida gerará uma interrupção interna. Este tipo de interrupção passa o processamento para uma posição fixa de memória. Se existir um programa supervisionando o sistema deve-se deixar a cargo do supervisor as providências a serem tomadas devido a violação de memória. Quando o processador estiver em estado supervisor pode-se ajustar as bases e os limites de maneira a permitir acesso a toda área de armazenamento.

O registrador base possui 16 bits, podendo apontar para qualquer posição da memória primária. O registrador limite possui também 16 bits, portanto os segmentos de área de armazenamento disponíveis podem ter até 64K palavras de memória.

2. Divisão lógica do espaço de endereçamento

Uma área disponível para armazenamento no processador é dividida logicamente em duas partes: Área de Dados e Área de Códigos (programa).

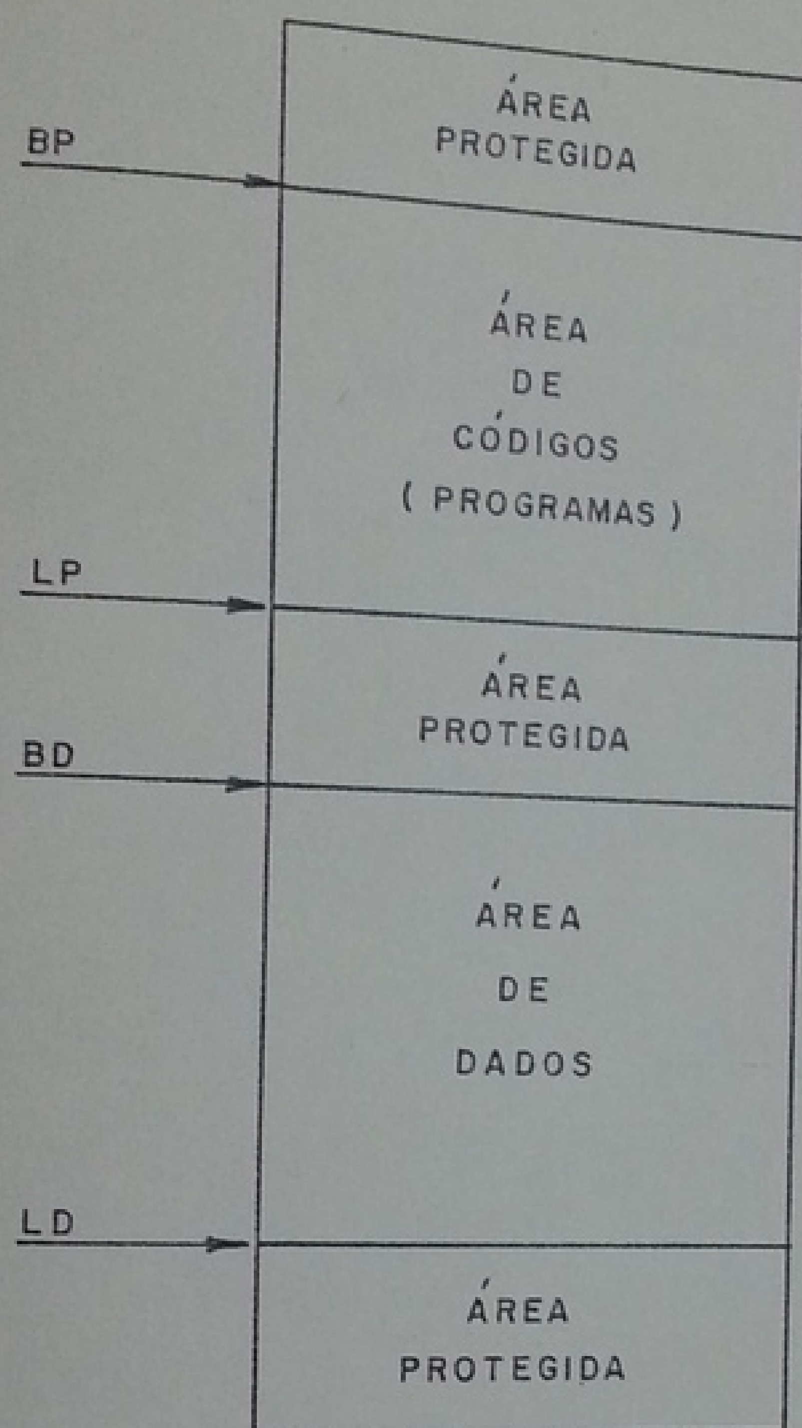
Esta divisão é feita por dois pares de ponteiros:

BASE - Limite de Dados

BASE - Limite de Programa

A área de dados pode ser modificada por um processo, caracterizando-se como uma área de trabalho, enquanto que a área de códigos não é alterável por um processo. Sendo assim, um segmento de código é fixo quando em execução, podendo ser partilhado por vários processos, não sendo necessário reproduzí-lo em diversas partes da memória. Portanto, os segmentos de códigos do processador terão caráter re-entrante.

Pode-se dizer que existe dentro de uma área permissível de memória, um outro nível de proteção pois a área de códigos é do tipo Ler-somente. Existem instruções privilegiadas, que manipulam as bases permitindo desta forma escrever em qualquer parte da memória. Este tipo de procedimento só é permitido no estado supervisor.



BP - BASE DE PROGRAMA

LP - LIMITE DE PROGRAMA

BD - BASE DE DADOS

LD - LIMITE DE DADOS

Os ponteiros BP e BD são usados para o cálculo de endereço efetivo, sendo acessados apenas por instruções privilegiadas e pelo controle do P.C.

3. Modos de Endereçamento

O processador possui um esquema de endereçamento basicamente relativo. As bases usadas para o cálculo de endereço efetivo são BP e BD. As instruções de máquina que possuem operando, especificam através do código de operação, qual das bases deve ser usada no cálculo do endereço efetivo do operando.

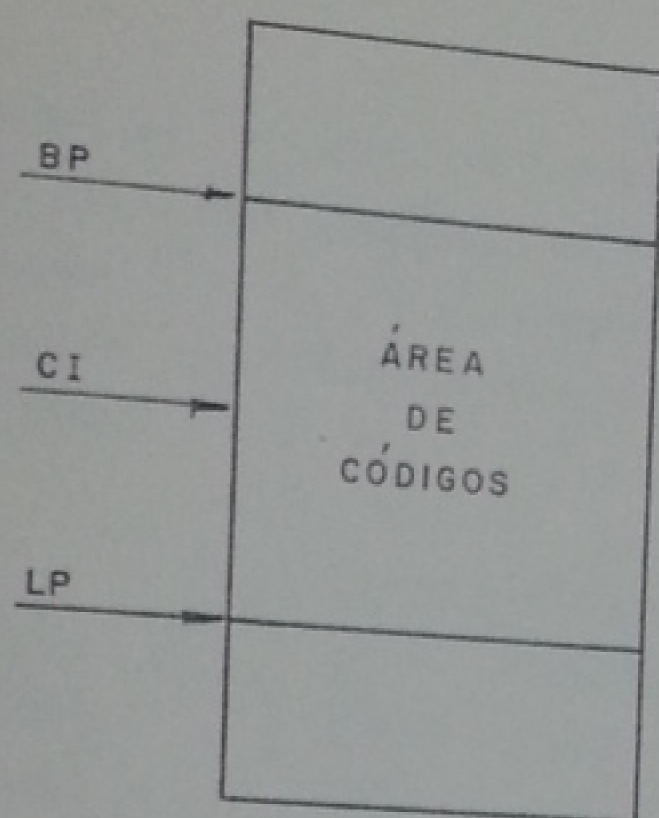
Sendo assim, toda instrução de referência à memória traz especificado no seu campo de endereço, um deslocamento relativo a uma das bases. Mesmo as referências à memória através dos registradores de propósito geral (oito registradores da rede)especificam um deslocamento relativo a uma das bases.

Com este esquema de endereçamento, as operações de relocação de segmentos de códigos e de dados, tornam-se extremamente simples, bastando alterar convenientemente as base de programa e de dados.

Quando o processador se encontra no estado supervisor a base de programa aponta para a posição zero da memória.

A amplitude dos campos de deslocamento é de 16 bits,possibilitando um acesso até 64K palavras de memória primária, tanto no esquema relativo como no esquema direto.

Para apontar a instrução a ser executada, existe na área de códigos um ponteiro contador de instruções (CI) cujo conteúdo é relativo à base de programa.



Portanto o endereço efetivo da próxima instrução é obtido pela seguinte expressão:

End.Efet.PINST = BP + CI + 1

Associado ao esquema relativo, pode-se ter ainda referências à memória indireta, indexada e através de regis tradores.

As instruções do processador podem apresentar até dois campos de endereço para referência à memória primária:na segunda palavra das instruções longas ou nos registradores de propósito geral. Para cada uma das possibilidades tem-se na instrução bits que determinam qual o procedimento para o cál culo de endereço efetivo que deve ser usado. Em ambos os ca sos o endereço que a instrução traz consigo é relativo a uma das bases (Programa ou dados) dependendo da própria instrução.

3.1. Procedimento de cálculo de endereço efetivo para um deslocamento na segunda palavra das instruções longas

As instruções que tem essa possibilidade de endereçamento possuem um campo chamado modo de endereçamento (ME) e outro de indexação (X) que determinam o processo de cálculo de endereço efetivo. Existem quatro possibilidades especificadas pelo campo ME:

- DIRETO /Pré-Indexado
- INDIRETO 1 nível /Pré-Indexado
- INDIRETO 2 nível /Pré-Indexado
- INDIRETO 1 nível /Pós-Indexado

A indexação vem especificada no campo X, que associado ao campo ME determina se um dos quatro procedimentos acima é indexado e qual registrador deve ser usado como índice.

<u>DIRETO</u>: Quando o campo ME especifica o modo direto o endereço efetivo é obtido simplesmente adicionando-se o conteúdo da segunda palavra da instrução à base de programa ou de dados dependendo da instrução.

A segunda palavra da instrução traz um deslocamento sempre positivo relativo a uma das bases.

Exemplo: CD, RØ, 1ØØ: Carregar o registrador RØ com o conteúdo da posição 1ØØ relativo à base de dados: .

RØ [23] antes

RØ [75] depois

BD = 501 → [ ]

601 → [75]

[ ]

EE = 501 + 100 = 601

portanto: (601) → RØ

EE = B + DESLOCAMENTO

EE → Endereço Efetivo

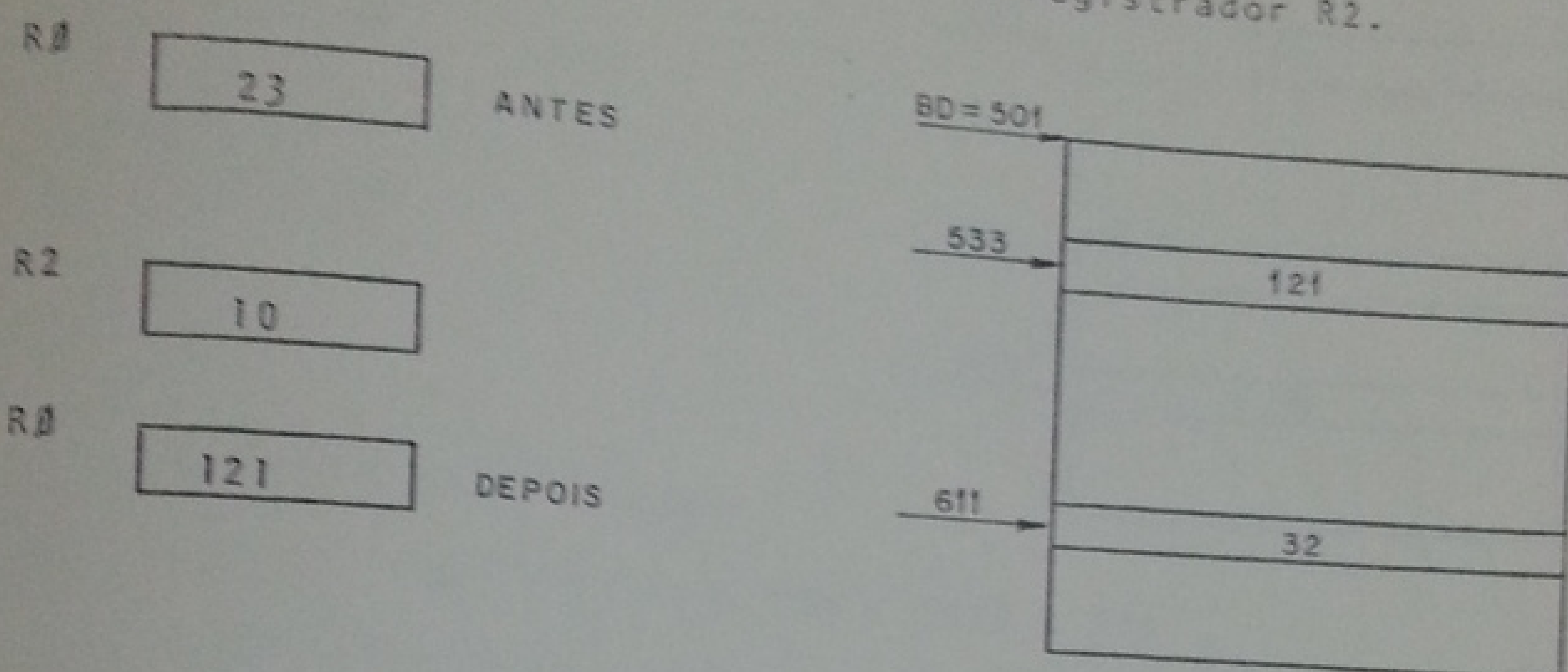
INDIRETO 1 NÍVEL PRÉ-INDEXADO: Se no campo ME estiver especificado este código sabe-se que a referência é indireta. Para saber se é indexado deve-se olhar o campo de indexação (X). Se houver indexação, ela precederá o acesso a memória. O campo X indica qual o registrador que deve ser usado como índice.

EE " (B + DESLOCAMENTO + RX) + B

Exemplo:

CD,RØ, I1, PRE, R x 2, 1ØØ

Esta instrução deve carregar o registrador RØ com o conteúdo da posição 1ØØ relativa a base de dados, referenciada indiretamente e pré-indexada com o registrador R2.



EE = (501 + 100 + 10) + 501 = (611) + 501 = 32 + 501
EE = 533 (533) ⟶ RØ

OBS.: (611) - conteúdo da posição 611 da memória primária

INDIRETO 2 NÍVEIS PRÉ-INDEXADO: Quando estiver especificado este tipo de procedimento no campo ME deve-se calcular o endereço efetivo da mesma maneira que no caso anterior (I1-PRÉ) simplesmente fazendo mais um nível de endereçamento indireto. Portanto a expressão simbólica fica sendo:

EE = ((B + DESLOCAMENTO + RX) + B) + B

Se não for especificado a indexação basta não somar o registrador de índice para obter o endereço efetivo:

Exemplo; CD, RØ, 12, R x 2, 1ØØ

RØ

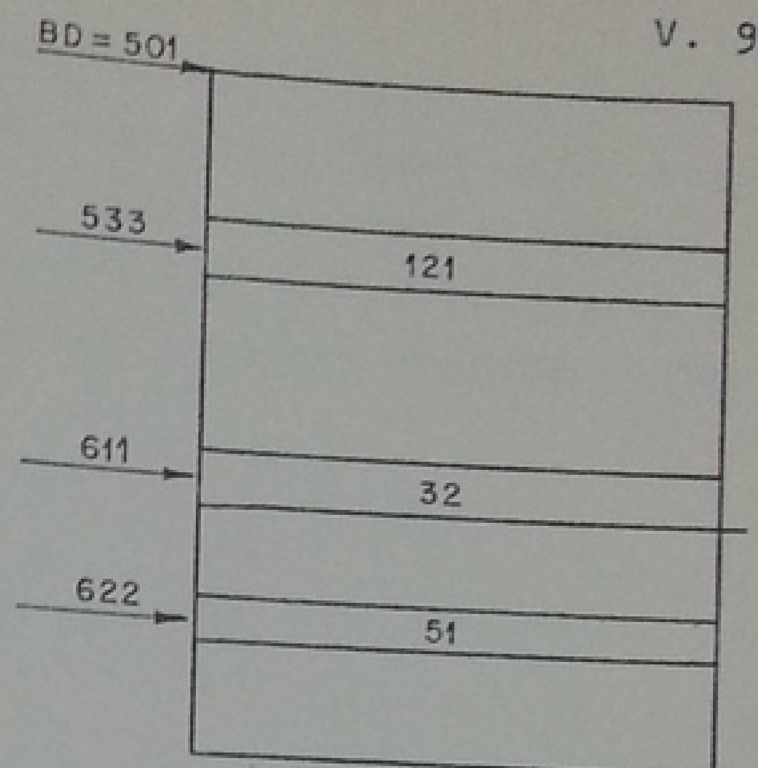
23

R2

1Ø

RØ

51



EE = (501+100+10)+501)+501

EE = ((611) + 501) + 501 = (32 + 501) + 501

EE = (533) + 501 = 121 + 501 EE = 622

(622) → RØ

<u>INDIRETO 1 NÍVEL PÓS-INDEXADO:</u> Para este tipo de procedimento a diferença está na hora em que se deve somar o registrador de índice. Se não for especificado indexação este modo se torna idêntico ao I1 pré-indexado.

EE = (B + DESLOCAMENTO) + RX + B

Exemplo: CD, RØ, I1, PÓS R X2, 1ØØ

RØ

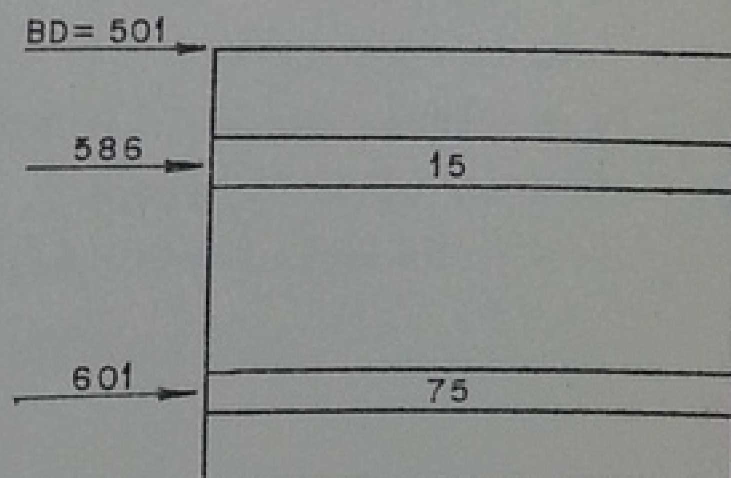
23

R2

10

RØ

15



EE = (501+100) + 10 + 501 = (601) + 10 + 501

EE = 75 + 10 + 501 = 586

(586) → RØ

As instruções do processador podem fazer referências à área de dados ou a área de programas. Todos os exemplos dados foram feitos na área de dados portanto obteve-se a fórmula de cálculo, da expressão simbólica substituindo-se B por BD. Para as referências à área de programa deve-se ter as seguintes expressões: .

DIRETO:

EE = BP + DESLOCAMENTO + RX

INDIRETO 1 NÍVEL PRÉ-INDEXADO

EE = (BP + DESLOCAMENTO + RX) + BP

INDIRETO 2 NÍVEIS PRÉ-INDEXADO

EE = ((BP + DESLOCAMENTO + RX) + BP) + BP

INDIRETO 1 NÍVEL PÓS-INDEXADO

EE = (BP + DESLOCAMENTO) + RX + BP

3.2. Procedimento de cálculo de endereço para um deslocamento contido num dos registradores de propósito geral:

Quando especifica-se um dos registradores de propósito geral através de uma instrução, dize-se também qual o modo de operação do registrador. A especificação de modo de operação vem num campo da instrução denominado MO.

Os diversos modos de operação possíveis são:

DIRETO (operando)

No campo MO vem especificado que o registrador referenciado pela instrução é o próprio operando.

Exemplo: INCR RØ

RØ [20] antes RØ [21] depois

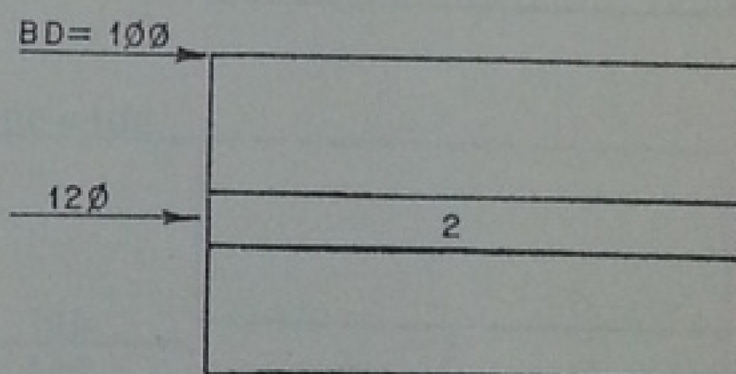
INDIRETO

Neste caso o registrador referenciado pela instrução é usado como ponteiro para uma posição de memória. O conteúdo do registrador é um deslocamento em relação a uma base.

EE = B + RN

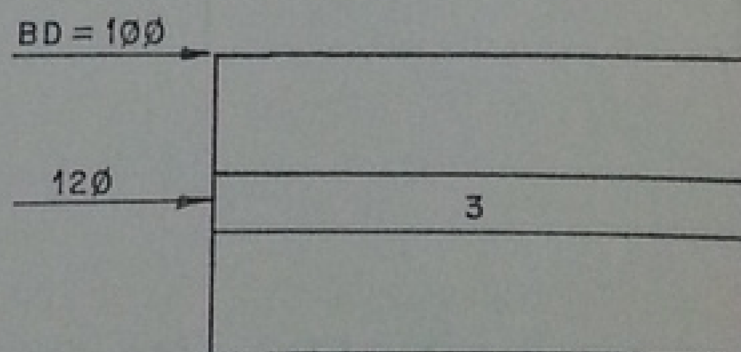
Exemplo: INCR RØ, I antes

RØ [2Ø]



DEPOIS

[2Ø]



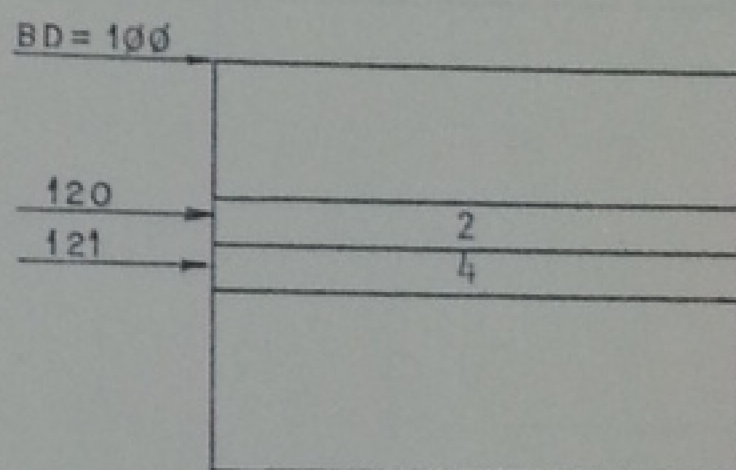
INDIRETO INCREMENTADO

Neste caso o registrador referenciado pela instrução é usado como ponteiro. Antes do cálculo do endereço efetivo e da busca do operando deve-se incrementar o registrador referenciado. Neste caso também o ponteiro é relativo a uma das bases.

Exemplo: (RN) + 1 → RN, EE = B + RN

DCR RØ, +1

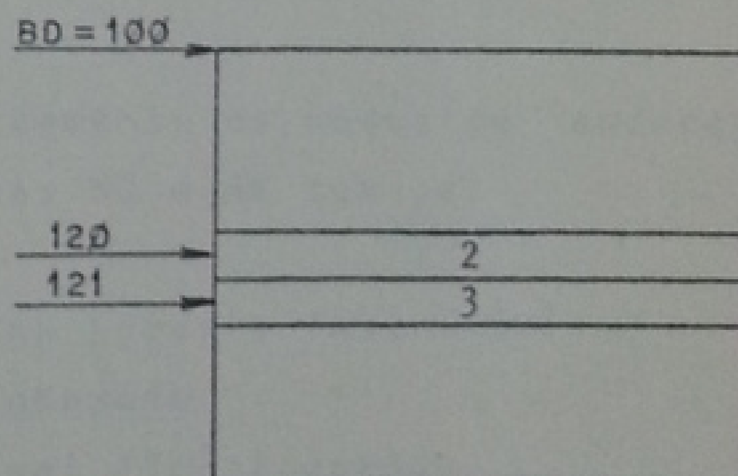
RØ [20]



EE = 1ØØ + 2Ø +1 = 121

(121) - 1 → (121)

RØ [21]

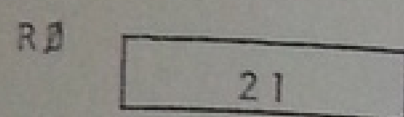


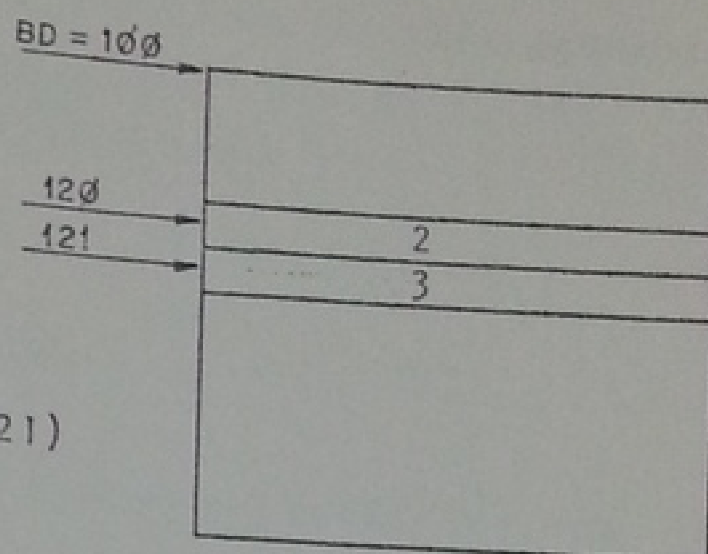
INDIRETO DECREMENTADO

O registrador referenciado pela instrução é decrementado depois de ser usado como ponteiro. O conteúdo do registrador é relativo a uma das bases.

$EE = B + RN, (RN) - 1 \rightarrow RN$

Exemplo: INCR RØ, 1

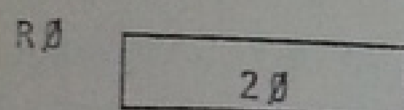


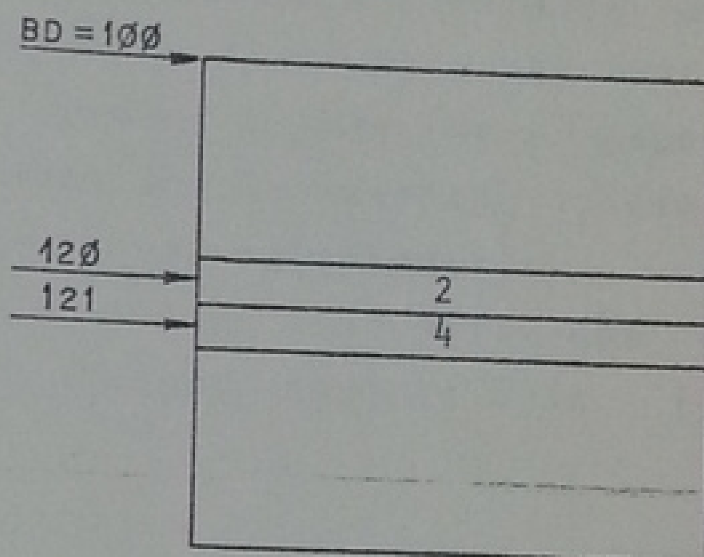


EE=1ØØ+21 = 121

(121) + 1 (121)

RØ - 1 → RØ





Resumindo-se esquematicamente os modos de endereçamento possíveis através dos campos MO e ME tem-se:

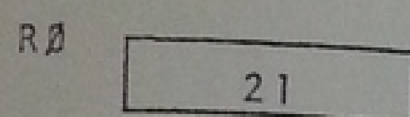
ME	(Deslocamento contido na própria instrução)
00	Direto Relativo/ Pré-indexado
01	Indireto Relativo 1 Nível /Pré-indexado
10	Indireto Relativo 2 Níveis /Pré-indexado
11	Indireto Relativo 1 Nível /Pós-indexado

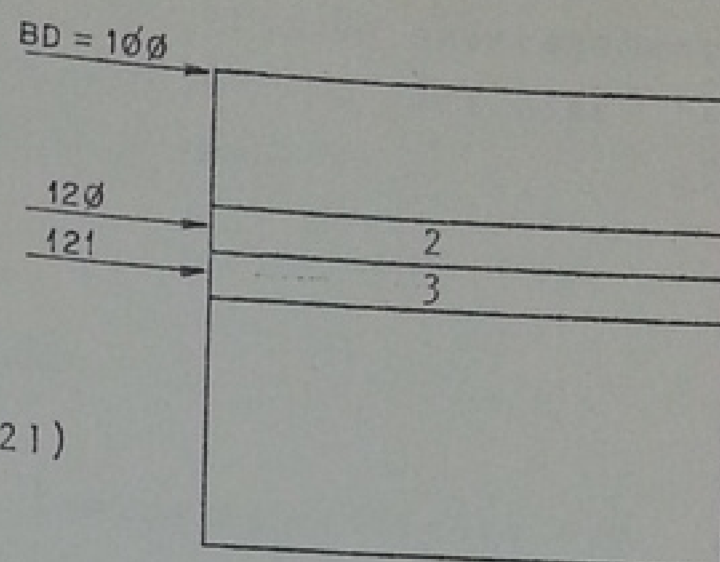
MO	(Referência a registrador)
00	Direto (operando)
01	Indireto Relativo
10	Indireto Relativo Pré-incrmentado
11	Indireto Relativo Pós-decrementado

EE = B + RN, (RN) - 1 → RN

Exemplo: INCR RØ, 1

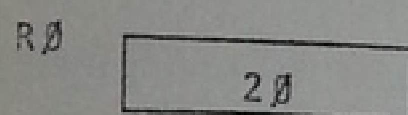
RØ

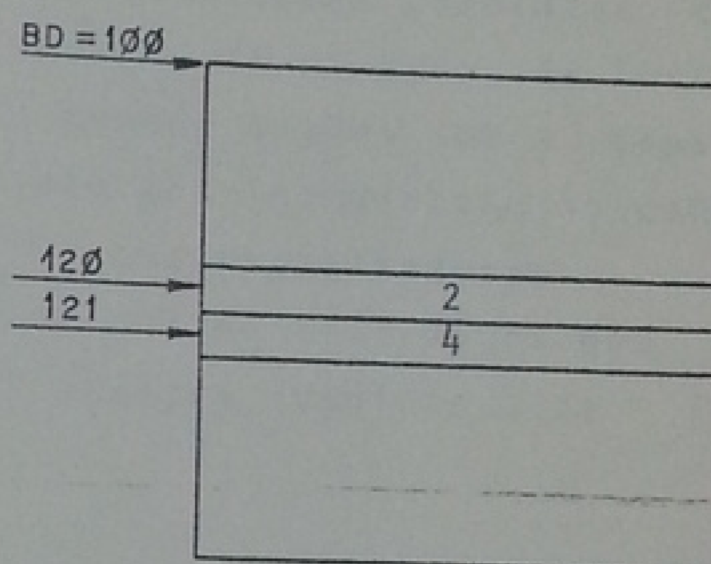




EE=1ØØ+21 = 121
(121) + 1 (121)
RØ - 1 → RØ

RØ





Resumindo-se esquematicamente os modos de endereçamento possíveis através dos campos MO e ME tem-se:

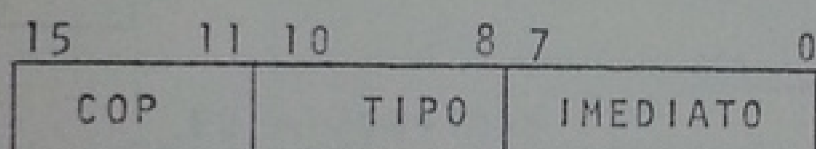
ME	(Deslocamento contido na própria instrução)
00	Direto Relativo/ Pré-indexado
01	Indireto Relativo 1 Nível /Pré-indexado
10	Indireto Relativo 2 Níveis /Pré-indexado
11	Indireto Relativo 1 Nível /Pós-indexado

MO	(Referência a registrador)
00	Direto (operando)
01	Indireto Relativo
10	Indireto Relativo Pré-incrmentado
11	Indireto Relativo Pós-decrementado

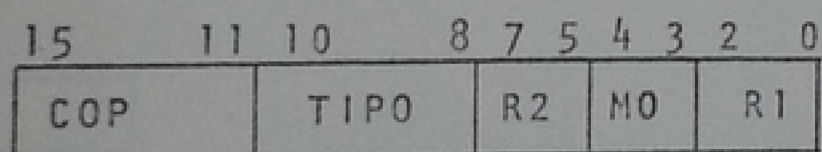
Além dos modos de endereçamento especificados pelos campos MO e ME das instruções de máquina, o processador possui ainda algumas características especiais de endereçamento que são utilizadas por instruções específicas para esse fim. Estas características são descritas a seguir.

- Endereçamento imediato

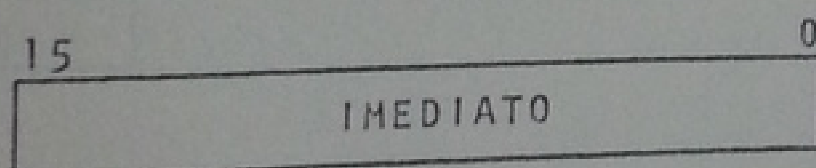
Existem instruções que trazem especificado num de seus campos um operando imediato. Este operando imediato pode ser de um byte ou de dois bytes. As instruções imediatas curtas trazem consigo um operando imediato de um byte. As instruções do tipo entre registradores, quando usadas com a especificação de RØ no campo R1 e indireto pré-incrmentado no campo MO, proporcionam operações entre um dos registradores de próprosito geral e um operando imediato de dois bytes que se encontra na área de programas, na posição seguinte à da instrução.



Imediatas curtas



Se MO = +1, R1 ≡ RØ(CI)



Imediatas longas

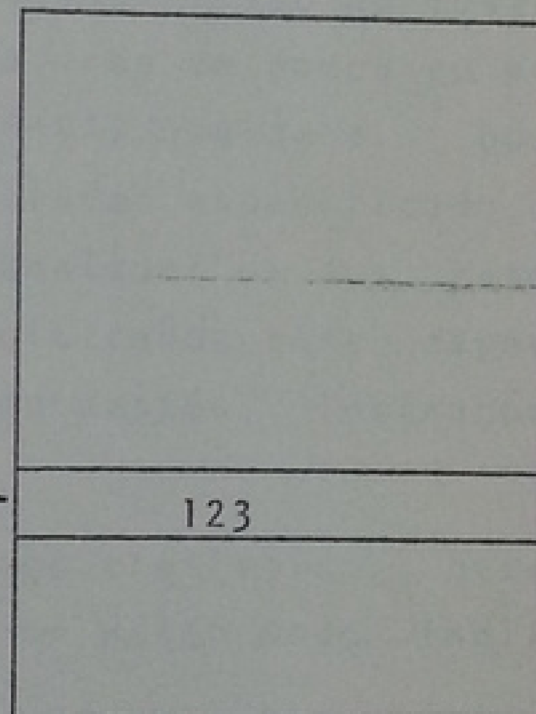
- Endereçamento relativo com deslocamento curto

Existem instruções curtas de referência à memória que possuem como campo de endereço um deslocamento de oito bits ($-128 \leq$ DESLOCAMENTO $\leq +127$). Estas instruções só fazem referências à área de dados e ainda assim, o deslocamento que elas trazem é relativo a R2. Como pode-se perceber estas instruções são relativas em dois níveis, uma vez em relação a R2 e outra vez em relação à base de dados já que o conteúdo de R2 é relativo a esta base.

Exemplo:

CD, R4, 10 BD = 200

Área de Dados



R2 [5]

antes

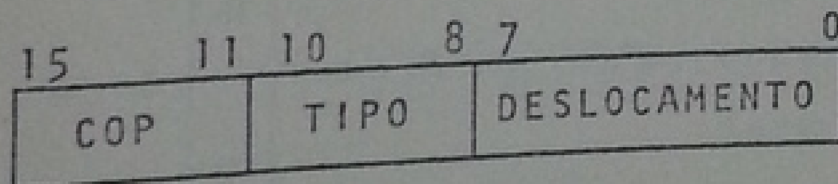
R4 [45]

EE = 200+10+5
(215)→R4

R2 [5]

depois

R4 [123]



EE = BD + R2 + DESLOCAMENTO

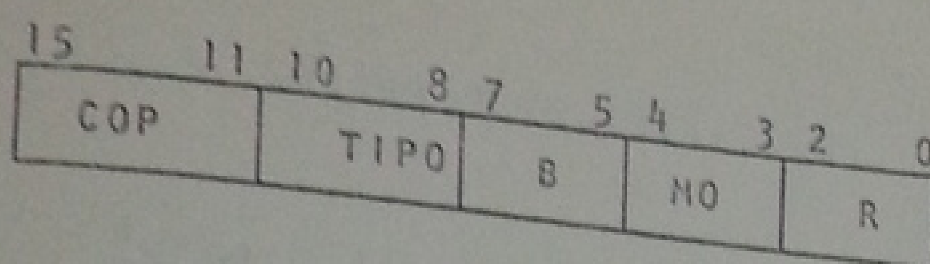
- Endereçamento de bytes

Existe no conjunto de instruções do processador um grupo de instruções que endereçam a memória primária byte a byte. Estas instruções são de uma palavra (curtas) e fazem referência a 2 registradores de propósito geral. Um deles é subespecificado pelo campo MO, podendo ser manipulados segundo uma das quatro alternativas fornecidas por este campo. Se for usado o modo direto a operação é efetuada sôbre os bytes direitos do registrador especificado e R7, não levando-se em conta o segundo registrador especificado na instrução. Se for usado um modo indireto o registrador especificado é encarado como um ponteiro cujo conteúdo é um deslocamento expresso em número de bytes. O segundo registrador especificado na instrução é usado como uma nova base dentro da área de dados ou programas. Se o registrador ponteiro for RØ(CI) usa-se a base de programas, para qualquer outro registrador especificado como ponteiro usa-se a base de dados. Em qualquer um dos casos o registrador ponteiro é relativo ao registrador base especificado pela instrução. O segundo operando destas instruções é sempre o byte direito do R7.

A fórmula de cálculo do endereço efetivo do byte que vai participar da operação, quando for usado modo indireto é a seguinte.

$$P/2 \Longrightarrow \begin{cases} \text{Resto} = 0 & \text{byte direito} \\ \text{Resto} = 1 & \text{byte esquerdo} \end{cases}$$

$$E\ E = B\begin{Bmatrix} D \\ \\ P \end{Bmatrix} + B + P/2$$ endereço da palavra que deve ser lida.

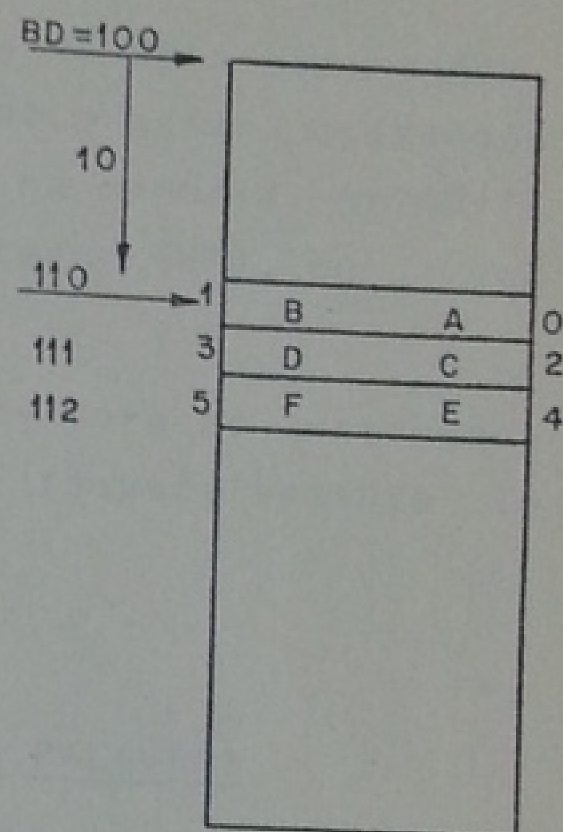


Exemplos: CD, R3, I R4

R3 [10]

R4 [3] antes

R7 [X | Y]



P = R4 3/2 = 1, RESTO = 1

EE = BD + R3 + R4/2

EE = 100 + 10 + 1 = 111

o byte procurado é o byte esquerdo da posição 111

R3 [10]

R4 [3] depois

R7 [X | D]

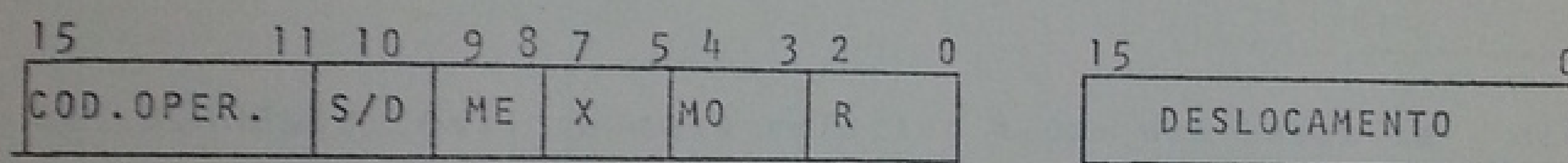
Como pode-se ver através do exemplo acima, a base especificada no campo B dessas instruções determina a origem de um sub-espeaço de endereçamento onde a unidade de informação manipulável é um byte.

- Formato das instruções

O processador possui instruções de dois operandos, de um operando ou sem operando. Em relação a esses operandos as instruções podem manipular dois dados na memória primária (MM), ou um na memória e outro em registrador (MR), ou os dois em registradores (RR).

Existem sete formatos principais para as instruções do processador, com uma ou duas palavras (respectivamente 16 ou 32 bits) que são descritos a seguir:

1º REFERÊNCIA A MEMÓRIA DE DUAS PALAVRAS



No qual os campos significam o seguinte:

COD.OPR: - Código de Operação, especificação da instrução

S/D: - Tipo de Operando manipulado por esta instrução

0 - Simples de 16 bits

1 - Duplo de 32 bits

ME: Modo de Endereçamento (referente ao operando especificado pelo deslocamento).

00	Direto
01	Indireto 1 nível /pré-indexação
10	Indireto 2 níveis /pré-indexação
11	Indireto 1 nível /pós-indexado

Nos dois casos (01,10), quando for especificada a indexação no campo X, ela ocorrerá antes da fase de indireto (pré-indexação).

No caso (11) se for especificado indexação ela ocorrerá após a fase de indireto (pós-indexação).

R: - Registrador

000 - Registrador 0 (contador de instruções)
001 - Registrador 1
010 - Registrador 2
011 - Registrador 3
100 - Registrador 4
101 - Registrador 5
110 - Registrador 6
111 - Registrador 7

DESLOCAMENTO: Deslocamento relativo a uma das bases de acordo com a instrução especificada (base de programa, base de dados).

MO: Modo de Operação do registrador especificado no campo R 0:2

00 Direto
01 Indireto
10 Indireto Incrementado
11 Indireto Decrementado

X: Indexação

000 - Não indexado
001 Indexado com o conteúdo do registrador R1
010 Indexado com o conteúdo do registrador R2
011 Indexado com o conteúdo do registrador R3
100 Indexado com o conteúdo do registrador R4
101 Indexado com o conteúdo do registrador R5

110 Indexado com o conteúdo do registrador R6
111 Indexado com o conteúdo do registrador R7

2º OPERAÇÕES ENTRE REGISTRADORES E MANIPULAÇÃO DE BYTES

CÓDIGO DE OPERAÇÃO: Como no formato anterior

TIPO: Sub-especificação da operação

R2: Registrador

000	Contador de instruções
001	Registrador 1
010	Registrador 2
011	Registrador 3
100	Registrador 4
101	Registrador 5
110	Registrador 6
111	Registrador 7

MO: Modo de operação referente ao registrador especificado no campo R1, como no formato anterior.

R1: Registrador

000	Contador de instruções
001	Registrador 1
010	Registrador 2
011	Registrador 3
100	Registrador 4
101	Registrador 5
110	Registrador 6
111	Registrador 7

OBSERVAÇÃO: Uma instrução indireta em relação ao registrador especificado pelo campo R1 torna-se uma instrução de operação entre 1 registrador e uma posição de memória

3º DESLOCAMENTOS E GIROS

15 – 11	10 – 8	7	6 – 3	2 – 0
COD	TIPO	S/D	NO	R1

COD. OPER.: Código de operação

S/D: Dado em simples ou dupla precisão

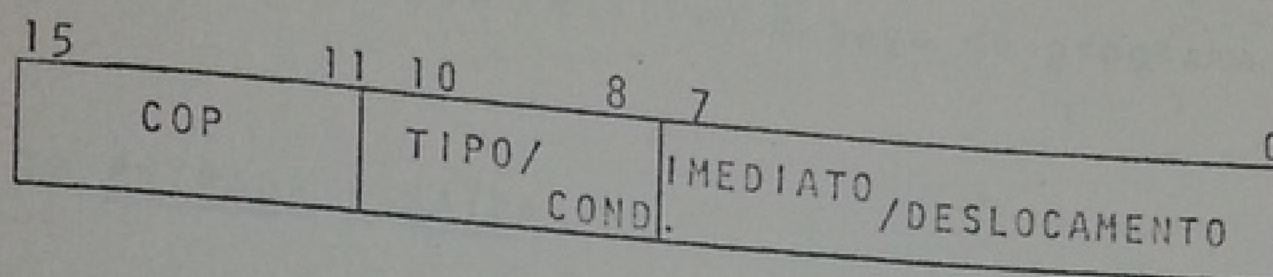
TIPO: Sub-especificação da instrução

Nº: Número de posições deslocadas

R1: Registrador ao qual se refere a operação especificada

000	Contador de instruções
001	Registrador 1
010	Registrador 2
011	Registrador 3
100	Registrador 4
101	Registrador 5
110	Registrador 6
111	Registrador 7

4º Imediatas curtas, Operações com deslocamentos curto, Modifica e testa status, Especiais curta.

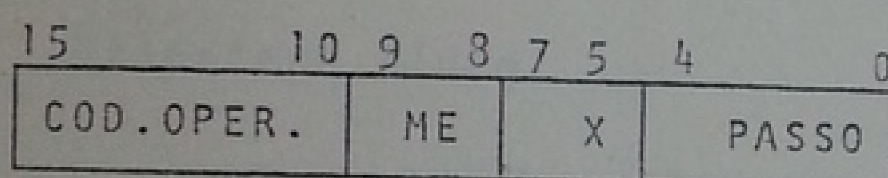


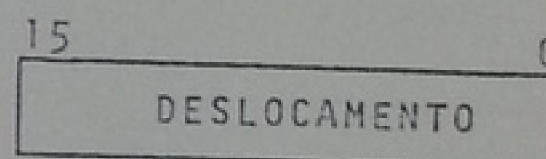
COP: Código de operação

TIPO/COND: Sub-especificação da instrução

Campo 0:7 : Operando imediato ou deslocamento curto

5º ESPECIAIS LONGAS





COD.OPER: Especificação da instrução

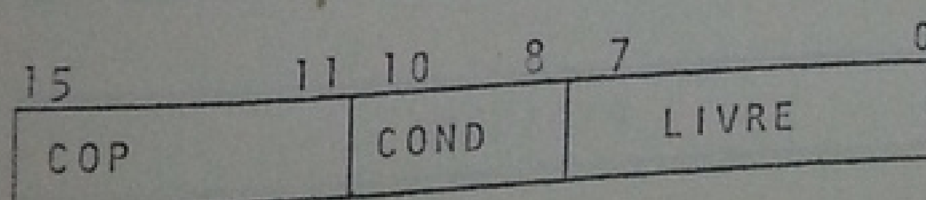
ME: Modo de endereçamento
Como nos casos anteriores

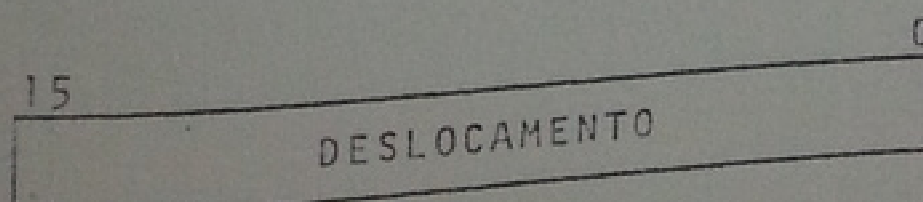
X: Indexação
Como nos casos anteriores

PASSO: Número binário ($-16 \leq$ passo ≤ 15)

DESLOCAMENTO: Deslocamento relativo à base de programa

6º PULO LONGO CONDICIONAL



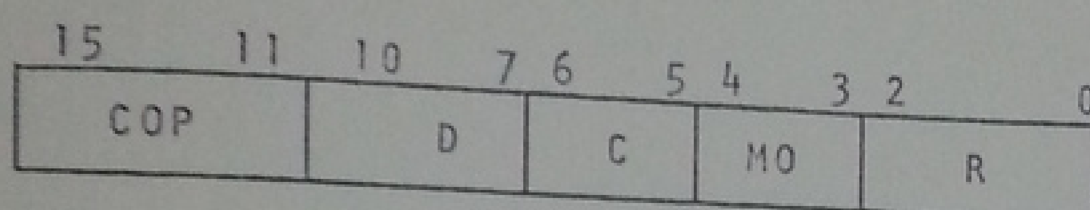


COP: Especificação da instrução

COND: Especifica a condição que deve ser verificada para provocar a execução da instrução

DESLOCAMENTO: Deslocamento relativo à base de programa

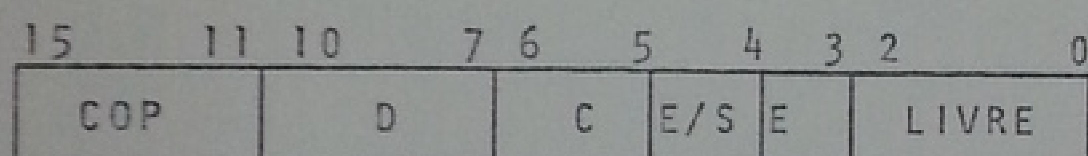
7º ENTRADA E SAÍDA



D: endereço do dispositivo

C : endereço do canal de E/S

. Transfere Bloco



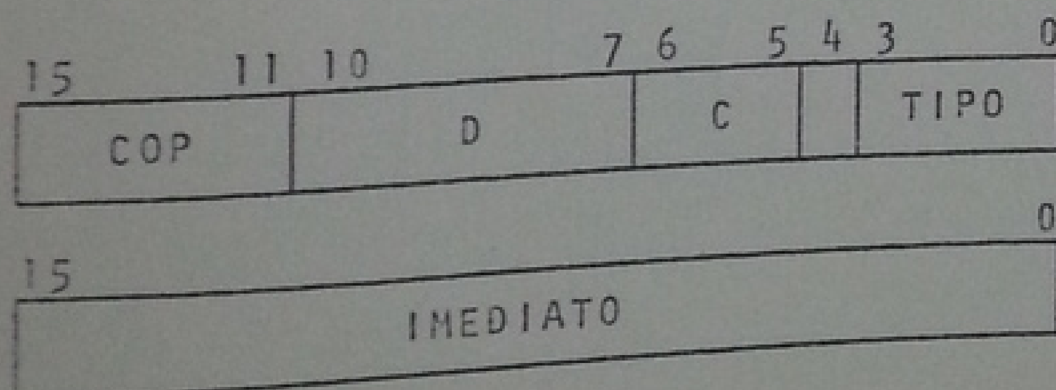
D: endereço do dispositivo

C: endereço do canal

E/S: indica se a transferência é de entrada ou de saída

E: indica se a transferência é com ou sem empacotamento.

. Função



D: endereço do dispositivo

C: endereço do canal

TIPO: sub-especificação da instrução

INSTRUÇÕES DO G-10*

INSTRUÇÕES	MNEMÔNICOS			FORMATOS (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)	(15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
ALTERA STATUS	XALT	NDISP, NCAN	CONST	0 0 1 0 1 / NDISP / NCAN / 0 / 1 0 0 1	CONSTANTE
ARMAZENA BYTE	ARMB	RBASE	$R_i \{,I_k\}$	0 1 1 0 1 / 1 1 1 / RBASE / MO / R_i	
ARMAZENA REG.	ARMR	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 1 1 1 / R_j / MO / R_i	
ARMAZENA BYTE da DIR. na DIR.	MVD	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 0 0 0 / R_j / MO / R_i	
ARMAZENA BYTE da DIR. na ESQ.	MVE	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 0 0 1 / R_j / MO / R_i	
ARMAZENA ENDEREÇO DE E/S	XEND	NDISP, NCAN	ENDEREÇO	0 0 1 0 1 / NDISP / NCAN / ø / 0 0 0 1	ENDEREÇO
ARMAZENA BYTE da ESQ. no BYTE da DIR. e LIMPA	SEP	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 0 1 0 / R_j / MO / R_i	
ARMAZENA LONGO	ARM	$R_j \{,I_k\}$	DESLOC $\{,R_i\} \{,I_l\}$	1 1 1 0 0 / S/D / ME / R_i / MO / R_j	DESLOCAMENTO
CARREGA CURTO	C	DESLOC		0 1 1 1 1 / 1 1 0 / DESLOCAMENTO	
CARREGA IMEDIATO	CARI	IMEDIATO		0 0 0 1 0 / 1 1 0 / OPERANDO IMEDIATO	
CARREGA BYTE	CB	RBASE	$R_i \{,I_k\}$	0 1 1 0 1 / 1 1 0 / RBASE / MO / R_i	
CARREGA BASE de DADOS	CBD	R_i		0 0 0 0 1 / 0 1 0 / 1 1 0 / 0 0 / R_i	
CARREGA BASE de PROGRAMA	CBP	R_i		0 0 0 0 1 / 0 1 0 / 1 1 1 / 0 0 / R_i	
CARREGA na ÁREA de DADOS LONGO	CD	$R_j \{,I_k\}$	DESLOC $\{,R_i\} \{,I_l\}$	1 1 1 1 0 / S/D / ME / R_i / MO / R_j	DESLOCAMENTO
CARREGA REG. na ÁREA de DADOS	CDR	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 1 0 1 / R_j / MO / R_i	
CARREGA LIMITE de DADOS	CLD	R_i		0 0 0 0 1 / 0 1 0 / 1 0 0 / 0 0 / R_i	
CARREGA LIMITE de PROGRAMA	CLP	R_i		0 0 0 0 1 / 0 1 0 / 1 0 1 / 0 0 / R_i	
COMPARA ARITM. C/ DESLOCAMENTO LONGO	CMP	$R_j \{,I_k\}$	DESLOC $\{,R_i\} \{,I_l\}$	1 0 1 0 0 / S/D / ME / R_i / MO / R_j	DESLOCAMENTO
COMPARA ARITM. DE BYTES	CPAB	RBASE	$R_i \{,I_k\}$	0 1 1 0 1 / 0 1 0 / RBASE / MO / R_i	
COMPARA ARITM. C/ DESLOCAMENTO CURTO	CMPC	DESLOC		0 1 1 1 1 / 0 1 0 / DESLOCAMENTO	
COMPARA IMEDIATO	CMPI	IMEDIATO		0 0 0 1 0 / 0 1 0 / OPERANDO IMEDIATO	
COMPARA REG. c/ ÁREA de PROGRAMA	CMPP	R_j	$R_i \{,I_k\}$	0 1 1 0 0 / 0 1 0 / R_j / MO / R_i	
CARREGA na ÁREA de PROGRAMA LONGO	CP	$R_j \{,I_k\}$	DESLOC $\{,R_i\} \{,I_l\}$	1 1 1 0 1 / S/D / ME / R_i / MO / R_j	DESLOCAMENTO
COMPLEMENTA	CPL	$R_i \{,I_k\}$		0 1 0 1 1 / 1 0 0 / 0 0 0 / MO / R_i	
CARREGA REG. na ÁREA de PROGRAMA	CPR	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 1 1 0 / R_j / MO / R_i	
CARREGA RELÓGIO	CRI	R_i		0 0 0 0 1 / 1 1 0 / 1 0 0 / 0 0 / R_i	
CHAMADA de SUPERVISOR	CSUP	OPERANDO		0 0 0 0 0 / 1 0 0 / OPERANDO	
COMPARA REG. c/ ÁREA de DADOS	CMPR	R_j	$R_i \{,J_k\}$	0 1 0 1 0 / 0 1 0 / R_j / MO / R_i	
COMPARA de BYTES ENTRE DOIS REGISTRADORES	CMPB	R_j	$R_i \{,I_k\}$	0 1 0 0 1 / 0 1 1 / R_j / MO / R_i	
DECREMENTA REG.	DCR	$R_i \{,I_k\}$		0 1 0 1 1 / 1 1 0 / 0 0 0 / MO / R_i	
DESLOCA LÓGICO A DIREITA SIMPLES	DLD	NDESLOC	R_i	0 0 0 1 1 / 0 0 0 0 / NDESLOC / R_i	
DESLOCA LÓGICO A DIREITA DUPLO	DLD	NDESLOC	R_i, D	0 0 0 1 1 / 0 0 0 1 / NDESLOC / R_i	
DESLOCA ARITM. A DIREITA SIMPLES	DAD	NDESLOC	R_i	0 0 0 1 1 / 0 1 0 0 / NDESLOC / R_i	
DESLOCA ARITM. A DIREITA DUPLO	DAD	NDESLOC	R_i, D	0 0 0 1 1 / 0 1 0 1 / NDESLOC / R_i	
DESLOCA LÓGICO A ESQ. SIMPLES	DLE	NDESLOC	R_i	0 0 0 1 1 / 0 0 1 0 / NDESLOC / R_i	
DESLOCA LÓGICO A ESQ. DUPLO	DLE	NDESLOC	R_i, D	0 0 0 1 1 / 0 0 1 1 / NDESLOC / R_i	
DESLOCA ARITM. A ESQ. SIMPLES	DAE	NDESLOC	R_i	0 0 0 1 1 / 0 1 1 0 / NDESLOC / R_i	
DESLOCA ARITM. A ESQ. DUPLO	DAE	NDESLOC	R_i, D	0 0 0 1 1 / 0 1 1 1 / NDESLOC / R_i	
DIVIDE C/ DESLOCAMENTO LONGO	DIV	$R_j \{,I_k\}$	DESLOC $\{,R_j\} \{,I_l\}$	1 1 0 0 0 / S/D / ME / R_i / MO / R_j	DESLOCAMENTO
ENTRADA DE DADOS	ENTR	NDISP, NCAN	$R_i \{,I_k\}$	0 0 1 1 0 / NDISP / NCAN / MO / R_i	
"E" C/ DESLOCAMENTO LONGO	E	$R_j \{,I_k\}$	DESLOC $\{,R_i\} \{,I_l\}$	1 0 1 1 1 / S/D / ME / R_i / MO / R_j	DESLOCAMENTO
EXECUTA na ÁREA de DADO	EXD	DESLOC $\{,R_j\} \{,I_l\}$		1 0 0 1 0 / 1 / ME / R_j / 0 0 / 0 0 0	DESLOCAMENTO
E ENTRE BYTES	EB	RBASE	$R_i \{,I_k\}$	0 1 1 0 1 / 0 1 1 / RBASE / MO / R_i	
E ENTRE REG. E ÁREA DE DADOS	ER	R_j	$R_i \{,I_k\}$	0 1 0 1 0 / 0 1 1 / R_j / MO / R_i	
E ENTRE REG. E ÁREA DE PROG.	EP	R_j	$R_i \{,I_k\}$	0 1 1 0 0 / 0 1 1 / R_j / MO / R_i	
E IMEDIATO CURTO	EI	IMEDIATO		0 0 0 1 0 0 1 1 / IMEDIATO	
ESPERE	ESP			0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	

INSTRUÇÕES DO G-10*

Instrução	Mnemônico	Operandos		15 14 13 12 11	10 9 8	7 6 5 4 3 2 1 0	15 ... 0 (2ª palavra)
EXECUTA DA ÁREA DO PROGRAMA	EXC	DESLOC [,R1] [,I2]		1 0 0 1 0	0 W E	R1 | 0 0 0 0 0	DESLOCAMENTO
E' COM DESLOCAMENTO CURTO	EE	DESLOC		0 1 1 1 1	0 1 1	DESLOCAMENTO	
ENTRA DE BLOCO EMPACOTADO	ENBP	NDISP, NCAN		0 0 1 0 0	NDISP	NCAN | 0 0 | 0 0 0	
ENTRA DE BLOCO DESEMPACOTADO	ENBD	NDISP, NCAN		0 0 1 0 0	NDISP	NCAN | 0 1 | 0 0 0	
FIM DE REPETE	FREP	R1		0 0 0 0 0	0 0 0	1 0 0 | 0 0 | R1	
FUNÇÃO DE RELÓGIO	FRE	OPERANDO		0 0 0 0 0	1 1 1	OPERANDO	
GIRA À DIREITA SIMPLES	GD	NGIROS	R1	0 0 0 1 1	1 0 0 0	NGIROS | R1	
GIRA À DIREITA DUPLO	[illegible]	NGIROS	R1,2	0 0 0 1 1	1 0 0 1	NGIROS | R1	
GIRA À ESQ. SIMPLES	[illegible]	NGIROS	R1	0 0 0 1 1	1 0 1 0	NGIROS | R1	
GIRA À ESQ. DUPLO	[illegible]	NGIROS	R1,2	0 0 0 1 1	1 0 1 1	NGIROS | R1	
INCREMENTA REG	INCR	R1 [,I2]		0 1 0 1 1	0 0 0	0 0 0 | W 0 | R1	
INIBE INTERRUPÇÃO	[illegible]	NDISP, NCAN		0 0 1 0 1	NDISP	NCAN | 0 | 0 1 1 1	
LER LIMITE DE DADOS	LLD	R1		0 0 0 0 1	0 0 0	1 0 0 | 0 0 | R1	
LER LIMITE DE PROG	LLP	R1		0 0 0 0 1	0 0 0	1 0 1 | 0 0 | R1	
LER BASE DE DADOS	LBD	R1		0 0 0 0 1	0 0 0	1 1 0 | 0 0 | R1	
LER BASE DE PROG	LBP	R1		0 0 0 0 1	0 0 0	1 1 1 | 0 0 | R1	
LER CHAVE	LCH	R1		0 0 0 0 1	0 0 1	1 0 0 | 0 0 | R1	
MOVE BYTE DA DIR. P/ DIR	MVD	R2	R1 [,I2]	0 1 0 0 1	0 0 0	R2 | W 0 | R1	
MOVE BYTE DA DIR. P/ ESQ	MVE	R2	R1 [,I2]	0 1 0 0 1	0 0 1	R2 | W 0 | R1	
MODIFICA MEMÓRIA	MOM	PASSO	DESLOC [,R1] [,I2]	1 0 0 1 1	1 W E	R1 | PASSO	DESLOCAMENTO
MODIFICA STATUS E/OU ENTRE A MÁSCARA E BYTE DIR	MST	0,0	MÁSCARA	0 1 0 0 0	1 0 0	MÁSCARA	
MODIFICA STATUS E/OU ENTRE A MÁSCARA E BYTE ESQ	MST	0,1	MÁSCARA	0 1 0 0 0	1 0 1	MÁSCARA	
MODIFICA STATUS E/E ENTRE A MÁSCARA E BYTE DIR	MST	1,0	MÁSCARA	0 1 0 0 0	1 1 0	MÁSCARA	
MODIFICA STATUS E/E ENTRE A MÁSCARA E BYTE ESQ	MST	1,1	MÁSCARA	0 1 0 0 0	1 1 1	MÁSCARA	
MULTIPLICA LONGO	M	R2 [,I2]	DESLOC [,R1] [,I2]	1 1 0 0 1	I2 W E	R1 | W 0 | R2	DESLOCAMENTO
"OU" C/ DESLOCAMENTO LONGO	OU	R2 [,I2]	DESLOC [,R1] [,I2]	1 0 1 1 0	I2 W E	R1 | W 0 | R2	DESLOCAMENTO
"OU" EXCLUSIVO C/ DESLOC. LONGO	OUE	R2 [,I2]	DESLOC [,R1] [,I2]	1 0 1 0 1	I2 W E	R1 | W 0 | R2	DESLOCAMENTO
"OU" ENTRE REG E ÁREA DE DADOS	OUR	R2	R1 [,I2]	[illegible]	1 0 0	R2 | W 0 | R1	
"OU" EXCLUSIVO ENTRE REG. E ÁREA DE DADOS	[illegible]	R2	R1 [,I2]	[illegible]	1 0 1	R2 | W 0 | R1	
"OU" ENTRE REG E ÁREA DE PROGRAMA	OUP	R2	R1 [,I2]	[illegible]	1 0 0	R2 | W 0 | R1	
"OU" EXCLUSIVO ENTRE REG. E ÁREA DE PROG	[illegible]	R2	R1 [,I2]	[illegible]	1 0 1	R2 | W 0 | R1	
"OU" IMEDIATO	OUI	IMEDIATO		[illegible]	1 0 0	IMEDIATO	
"OU" EXCLUSIVO IMEDIATO	[illegible]	IMEDIATO		[illegible]	1 0 1	IMEDIATO	
"OU" ENTRE BYTES	OUB	RBASE	R1 [,I2]	[illegible]	1 0 0	RBASE | W 0 | R1	
"OU" EXCLUSIVO C/ BYTE	[illegible]	RBASE	R1 [,I2]	[illegible]	1 0 1	RBASE | W 0 | R1	
"OU" C/ DESLOCAMENTO CURTO	[illegible]	DESLOC		0 1 1 1 1	1 0 0	DESLOCAMENTO	
"OU" EXCLUSIVO C/ DESLOCAMENTO CURTO	[illegible]	DESLOC		0 1 1 1 1	1 0 1	DESLOCAMENTO	

INSTRUÇÕES DO G-10*

Instrução	Mnemônico	Operandos		Bits 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	2ª palavra: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
EXECUTA NA ÁREA DO PROGRAMA	EXC	DESLOC. {, R_j} {, I_ℓ}		1 0 0 1 0 | 0 | M E | Rj | 0 0 0 0 0	DESLOCAMENTO
É COM DESLOCAMENTO CURTO	EC	DESLOC.		0 1 1 1 1 | 0 1 1 | DESLOCAMENTO	
ENTRA DE BLOCO EMPACOTADO	ENBP	NDISP, NCAN.		0 0 1 0 0 | NDISP | NCAN | 0 0 | 0 0 0	
ENTRA DE BLOCO DESEMPACOTADO	ENBD	NDISP, NCAN.		0 0 1 0 0 | NDISP | NCAN | 0 1 | 0 0 0	
FIM DE REPETE	FREP	R_i		0 0 0 0 0 | 0 0 0 | 1 0 0 | 0 0 | R_i	
FUNÇÃO DE RELÓGIO	FRI	OPERANDO		0 0 0 0 0 | 1 1 1 | OPERANDO	
GIRA À DIREITA SIMPLES	GD	NGIROS	R_i	0 0 0 1 1 | 1 0 0 0 | NGIROS | R_i	
GIRA À DIREITA DUPLO	GD	NGIROS	R_i, D	0 0 0 1 1 | 1 0 0 1 | NGIROS | R_i	
GIRA À ESQ. SIMPLES	GE	NGIROS	R_i	0 0 0 1 1 | 1 0 1 0 | NGIROS | R_i	
GIRA À ESQ. DUPLO	GE	NGIROS	R_i, D	0 0 0 1 1 | 1 0 1 1 | NGIROS | R_i	
INCREMENTA REG.	INCR	R_i {, I_k}		0 1 0 1 1 | 0 0 0 | 0 0 0 | M O | R_i	
INIBE INTERRUPÇÃO	XINI	NDISP, NCAN.		0 0 1 0 1 | NDISP | NCAN | 0 | 0 1 1 1	
LER LIMITE DE DADOS	LLD	R_i		0 0 0 0 1 | 0 0 0 | 1 0 0 | 0 0 | R_i	
LER LIMITE DE PROG.	LLP	R_i		0 0 0 0 1 | 0 0 0 | 1 0 1 | 0 0 | R_i	
LER BASE DE DADOS	LBD	R_i		0 0 0 0 1 | 0 0 0 | 1 1 0 | 0 0 | R_i	
LER BASE DE PROG.	LBP	R_i		0 0 0 0 1 | 0 0 0 | 1 1 1 | 0 0 | R_i	
LER CHAVE	LCH	R_i		0 0 0 0 1 | 0 0 1 | 1 0 0 | 0 0 | R_i	
MOVE BYTE DA DIR. P/ DIR.	MVD	R_j	R_i {, I_k}	0 1 0 0 1 | 0 0 0 | R_j | M O | R_i	
MOVE BYTE DA DIR. P/ ESQ.	MVE	R_j	R_i {, I_k}	0 1 0 0 1 | 0 0 1 | R_j | M O | R_i	
MODIFICA MEMÓRIA	MDM	PASSO	DESLOC. {, R_j} {, I_ℓ}	1 0 0 1 1 | 1 | M E | R_j | PASSO	DESLOCAMENTO
MODIFICA STATUS C/ 'OU' ENTRE A MÁSCARA E BYTE DIR.	MST	Q, D	MÁSCARA	0 1 0 0 0 | 1 0 0 | MÁSCARA	
MODIFICA STATUS C/ 'OU' ENTRE A MÁSCARA E BYTE ESQ.	MST	Q, E	MÁSCARA	0 1 0 0 0 | 1 0 1 | MÁSCARA	
MODIFICA STATUS C/ 'E' ENTRE A MÁSCARA E BYTE DIR.	MST	E, D	MÁSCARA	0 1 0 0 0 | 1 1 0 | MÁSCARA	
MODIFICA STATUS C/ 'E' ENTRE A MÁSCARA E BYTE ESQ.	MST	E, E	MÁSCARA	0 1 0 0 0 | 1 1 1 | MÁSCARA	
MULTIPLICA LONGA	M	R_j {, I_k}	DESLOC {, R_i} {, I_ℓ}	1 1 0 0 1 | S/D | M E | R_i | M O | R_j	DESLOCAMENTO
'OU' C/ DESLOCAMENTO LONGO	OU	R_j {, I_k}	DESLOC {, R_i} {, I_ℓ}	1 0 1 1 0 | S/L | M E | R_i | M O | R_j	DESLOCAMENTO
'OU' EXCLUSIVO C/ DESLOC. LONGO	OUX	R_j {, I_k}	DESLOC {, R_i} {, I_ℓ}	1 0 1 0 1 | S/D | M E | R_i | M O | R_j	DESLOCAMENTO
'OU' ENTRE REG. E ÁREA DE DADOS	OUR	R_j	R_i {, I_k}	0 1 0 1 0 | 1 0 0 | R_j | M O | R_i	
'OU' EXCLUSIVO ENTRE REG. E ÁREA DE DADOS	OUXR	R_j	R_i {, I_k}	0 1 0 1 0 | 1 0 1 | R_j | M O | R_i	
'OU' ENTRE REG. E ÁREA DE PROGRAMA	OUP	R_j	R_i {, I_k}	0 1 1 0 0 | 1 0 0 | R_j | M O | R_i	
'OU' EXCLUSIVO ENTRE REG. E ÁREA DE PROG.	OUXP	R_j	R_i {, I_k}	0 1 1 0 0 | 1 0 1 | R_j | M O | R_i	
'OU' IMEDIATO	OUI	IMEDIATO		0 0 0 1 0 | 1 0 0 | IMEDIATO	
'OU' EXCLUSIVO IMEDIATO	OUXI	IMEDIATO		0 0 0 1 0 | 1 0 1 | IMEDIATO	
'OU' ENTRE BYTES	OUB	RBASE	R_i {, I_k}	0 1 1 0 1 | 1 0 0 | RBASE | M O | R_i	
'OU' EXCLUSIVO C/ BYTE	OUXB	RBASE	R_i {, I_k}	0 1 1 0 1 | 1 0 1 | RBASE | M O | R_i	
'OU' C/ DESLOCAMENTO CURTO	OUC	DESLOC		0 1 1 1 1 | 1 0 0 | DESLOCAMENTO	
'OU' EXCLUSIVO C/ DESLOCAMENTO CURTO	OUXC	DESLOC		0 1 1 1 1 | 1 0 1 | DESLOCAMENTO	

INSTRUÇÕES DO G10*

Instrução	Mnemônico	Operandos		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
PARA	PARE			0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0	
PARA E/S	XPAR	N DISP, NCAN.		0 0 1 0 1 N DISP NCAN. 0 0 0 1 1	
PERMITE INTERRUPÇÃO E/S	XPER	N DISP, NCAN.		0 0 1 0 1 N DISP NCAN. 0 0 1 0 1	
PULA P/ SUBROTINA	PUG	Rj {,Ik}	DESLOC. {,Ri} {,Iℓ}	1 0 0 1 1 0 ME Ri MO Rj	DESLOCAMENTO
PULA C/ DESL. LONGO INCONDICIONAL	PLA	DESLOC.		1 0 0 0 1 0 0 0	DESLOCAMENTO
PULA C/ DESL. LONGO SE C=1	PLCL	DESLOC.		1 0 0 0 1 0 0 1	DESLOCAMENTO
PULA C/ DESL. LONGO SE ≤	PNZL	DESLOC.		1 0 0 0 1 0 1 0	DESLOCAMENTO
PULA C/ DESL. LONGO SE >	PPL	DESLOC.		1 0 0 0 1 0 1 1	DESLOCAMENTO
PULA C/ DESL. LONGO SE ≠	PDIL	DESLOC.		1 0 0 0 1 1 0 0	DESLOCAMENTO
PULA C/ DESL. LONGO SE =	PZL	DESLOC.		1 0 0 0 1 1 0 1	DESLOCAMENTO
PULA C/ DESL. LONGO SE ≥	PPZL	DESLOC.		1 0 0 0 1 1 1 0	DESLOCAMENTO
PULA C/ DESL. LONGO SE <	PNL	DESLOC.		1 0 0 0 1 1 1 1	DESLOCAMENTO
PULA C/ DESL. CURTO INCONDICIONAL	PLAC	DESLOC.		0 1 1 1 0 0 0 0 DESLOCAMENTO	
PULA C/ DESL. CURTO SE C=1	PLC	DESLOC.		0 1 1 1 0 0 0 1 DESLOCAMENTO	
PULA C/ DESL. CURTO SE ≤	PMEI	DESLOC.		0 1 1 1 0 0 1 0 DESLOCAMENTO	
PULA C/ DESL. CURTO SE >	PP	DESLOC.		0 1 1 1 0 0 1 1 DESLOCAMENTO	
PULA C/ DESL. CURTO SE ≠	PDI	DESLOC.		0 1 1 1 0 1 0 0 DESLOCAMENTO	
PULA C/ DESL. CURTO SE =	PI	DESLOC.		0 1 1 1 0 1 0 1 DESLOCAMENTO	
PULA C/ DESL. CURTO SE ≥	PMAI	DESLOC.		0 1 1 1 0 1 1 0 DESLOCAMENTO	
PULA C/ DESL. CURTO SE <	PN	DESLOC.		0 1 1 1 0 1 1 1 DESLOCAMENTO	

Instrução	Mnemônico	Operandos	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RESTAURA REGISTRADOR	RSTR	Ri	0 0 0 0 1 1 0 0 1 0 0 0 0 Ri
RESTAURA STATUS	RSTS	Ri	0 0 0 0 0 1 0 1 0 0 0 0 0 Ri
REPETE	REP		0 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1
RETORNA DE ITINTERRUPÇÃO	RIT	OPERANDO	0 0 0 0 0 1 1 0 0 0 C2 C1 C0 RI P 0

- P - LIBERA NÍVEIS DE IT DE PRIORIDADE MENOR QUE O PAINEL
- RI - LIBERA NÍVEIS DE IT DE PRIORIDADE MENOR QUE O RELÓGIO INTERNO
- C0 - LIBERA NÍVEIS DE IT DE PRIORIDADE MENOR QUE O CANAL 0 DE E/S
- C1 - LIBERA NÍVEIS DE IT DE PRIORIDADE MENOR QUE O CANAL 1 DE E/S
- C2 - LIBERA NÍVEIS DE IT DE PRIORIDADE MENOR QUE O CANAL 2 DE E/S

Instrução	Mnemônico	Operandos		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
SAÍDA DE BLOCO DESEMPACOTADA	SABD	NDISP, NCAN		0 0 1 0 0 NDISP NCAN 1 1 0 0 0	
SAÍDA DE BLOCO EMPACOTADA	SABP	NDISP, NCAN		0 0 1 0 0 NDISP NCAN 1 0 0 0 0	
SAÍDA DE DADOS	SAI	NDISP, NCAN	Ri {,Ik}	0 0 1 1 1 NDISP NCAN MO Ri	
SOMA LONGO	SOM	Rj {,Ik}	DESLOC {,Ri} {,Iℓ}	1 1 0 1 1 ½0 ME Ri MO Rj	DESLOCAMENTO
SUBTRAI LONGO	SUB	Rj {,Ik}	DESLOC {,Ri} {,Iℓ}	1 1 0 1 0 ½0 ME Ri MO Rj	DESLOCAMENTO
SOMA REG. C/ÁREA DE DADOS	SOMR	Rj	Ri {,Ik}	0 1 0 1 0 0 0 0 Rj MO Ri	
SOMA REG. C/ÁREA DE PROGRAMA	SOMP	Rj	Ri {,Ik}	0 1 1 0 0 0 0 0 Rj MO Ri	
SUBTRAI REG. C/ÁREA DE DADOS	SUBR	Rj	Ri {,Ik}	0 1 0 1 0 0 0 1 Rj MO Ri	
SUBTRAI REG. C/ÁREA DE PROGRAMA	SUBP	Rj	Ri {,Ik}	0 1 1 0 0 0 0 1 Rj MO Ri	
SALVA REGISTRADOR	SVR	Rj		0 0 0 0 1 0 1 1 1 0 0 0 0 Rj	
SALVA STATUS	SVS	Rj		0 0 0 0 0 0 1 1 0 0 0 0 0 Ri	
SOMA IMEDIATO	SOMI	IMEDIATO		0 0 0 1 0 0 0 0 IMEDIATO	
SUBTRAI IMEDIATO	SUBI	IMEDIATO		0 0 0 1 0 0 0 1 IMEDIATO	
SOMA BYTE	SOMB	R BASE	Ri {,Ik}	0 1 1 0 1 0 0 0 RBASE MO Ri	
SUBTRAI BYTE	SUBB	R BASE	Ri {,Ik}	0 1 1 0 1 0 0 1 RBASE MO Ri	
SOMA COM. DESL. CURTO	SOMC	DESLOC.		0 1 1 1 1 0 0 0 DESLOCAMENTO	
SUBTRAI COM. DESL. CURTO	SUBC	DESLOC.		0 1 1 1 1 0 0 1 DESLOCAMENTO	
SEPARA BYTE	SEP	Rj	Ri {,Ik}	0 1 0 0 1 0 1 0 Rj MO Ri	

INSTRUÇÕES DO G-10*

Instrução	Mnemônico	15 14 13 12 11	10 9 8	7 6 5	4 3	2 1 0	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TROCA MEMORIA C/MEMORIA	TR Rj{,Ik} DESLOC.{,r i}{,Iℓ}	1 1 1 1 1	P/D M E	Ri	M O	Rj	DESLOCAMENTO
TROCA SINAL	TRS Ri{,Ik}	0 1 0 1 1	0 1 1	0 0 0	M O	Ri	
TESTA STATUS C/MASC. E NO BYTE DR.	TST E,D MASCARA	0 1 0 0 0	0 0 0	MASCARA			
" E " " ESQ.	TST E,E MASCARA	0 1 0 0 0	0 0 1	MASCARA			
OU " " DIR.	TST O,D MASCARA	0 1 0 0 0	0 1 0	MASCARA			
OU " " ESQ.	TST O,E MASCARA	0 1 0 0 0	0 1 1	MASCARA			
TESTA VETOR DE INT. DE E/S	XTVT NDISP,NCANAL IMEDIATO	0 0 1 0 1	NDISP	C	0	1 0 1 0	IMEDIATO
" STATUS " E/S	XTES NDISP,NCANAL IMEDIATO	0 0 1 0 1	NDISP	C	0	1 1 0 0	IMEDIATO

Instrução	Mnemônico	15 14 13 12 11	10 9 8	7 6 5	4 3	2 1 0
ZERA REG.	ZER Ri{,Ik}	0 1 0 1 1	0 0 1	0 0 0	M O	Ri

CONVENÇÕES:

NCAN = Nº DO CANAL

NDISP = Nº DO DISPOSITIVO

0 ≤ j ≤ 7 — Ex: R5 = REGISTRADOR Nº5

0 ≤ i ≤ 7 — Ex: R3 = REGISTRADOR Nº3

1 ≤ ℓ ≤ 3 — ONDE:

I1 INDIRETO 1 NIVEL/PRÉ-INDEXADO

I2 INDIRETO 2 " /PRÉ-INDEXADO

I3 INDIRETO 1 " /PÓS-INDEXADO

1 ≤ K ≤ 3 — ONDE:

I1 - INDIRETO

I2 - INDIRETO INCREMENTADO

I3 - INDIRETO DECREMENTADO

OBS: 1 - NA INSTRUÇÃO PUG RECOMENDA-SE O USO DE k = 2 AFIM DE EVITAR PROBLEMAS C/A PILHA.

2 - INSTRUÇÃO REPETE: VIDE ARMAZENA REG.

3 - VETOR DE INTERRUPÇÃO:

15 14 13 12 11 10	9	8	7	6	5	4 3	2 1 0
	E	EC	F	0	0	NCAN	NDISP

- F — FIM DE TRANSFERÊNCIA
- EC — ERRO POR SUPERPOSIÇÃO DE INSTRUÇÃO
- E — ERRO NO DISPOSITIVO

4 - PALAVRA DE STATUS

OPERANDO DA CHAMADA OU SUB.	C	I	T	M	P/I	Z	S	E	T

- C — CONDIÇÃO
- I — INTERRUPÇÃO
- M — MODO
- P/I — PAR/IMPAR
- Z — ZERO
- S — SINAL
- E — EXTENSÃO
- T — TRANSBORDO

* OS MNEMÔNICOS SÃO VÁLIDOS PARA O MONTADOR DO SIMULADOR

CLASSES

15 14 13 12 11	CLASSES	BASE	TAMANHO	INDICE	PONTEIRO	REGISTRADOR	INDIRETO	OPERANDO	PRIVILEGIADA	Nº DE CICLOS
0 0 0 0 0	ESPECIAIS DE UM PALAVRA		C						/X	
0 0 0 0 1	ESPECIAIS COM REFERÊNCIA A REG.		C			X			/X	
0 0 0 1 0	IMEDIATAS CURTAS		C			X				
0 0 0 1 1	DESLOCAMENTOS E GIROS		C			X				
0 0 1 0 0	TRANSFERE BLOCO		C						X	
0 0 1 0 1	FUNÇÃO DE ENTRADA/SAIDA		L						X	
0 0 1 1 0	ENTRADA DE DADOS	D	C		X	X			X	
0 0 1 1 1	SAIDA DE DADOS	D	C		X	X			X	
0 1 0 0 0	MODIFICAÇÃO E TESTE DE STATUS		C						X	
0 1 0 0 1	ENTRE REG. CARGA E ARMAZENAGEM	P/D	C		X	X				
0 1 0 1 0	ENTRE REG. ARIT. E LOG. ÁREA DE DADOS	D	C		X	X				
0 1 0 1 1	MANIPULAÇÃO DE UM REGISTRADOR	D	C		X	X				
0 1 1 0 0	ENTR. E REG. ARIT. E LOG. ÁREA DE PROGRAMAS	P	C		X	X				
0 1 1 0 1	MANIPULAÇÃO DE BYTES	P/D	C		X	X				
0 1 1 1 0	PULOS CURTOS CONDICIONAIS	CI	C							
0 1 1 1 1	ARIT. E LOG. COM DESLOCAMENTO CURTO	D	C			X				
1 0 0 0 0										
1 0 0 0 1	PULOS LONGOS CONDICIONAIS	P	L							
1 0 0 1 0	EXECUTE	P/D	L	X			X			
1 0 0 1 1	PULO PARA SUBROTINA/MODIFICA E TESTA	P/D	L	X			X			
1 0 1 0 0	COMPARAÇÃO ARITMETICA	D	L	X	X	X	X			
1 0 1 0 1	OUX	D	L	X	X	X	X			
1 0 1 1 0	OU	D	L	X	X	X	X			
1 0 1 1 1	E	D	L	X	X	X	X			
1 1 0 0 0	DIVISÃO	D	L	X	X	X	X			
1 1 0 0 1	MULTIPLICAÇÃO	D	L	X	X	X	X			
1 1 0 1 0	SUBTRAÇÃO	D	L	X	X	X	X			
1 1 0 1 0	SOMA	D	L	X	X	X	X			
1 1 1 0 0	ARMAZENAGEM	D	L	X	X	X	X			
1 1 1 0 1	CARGA DA ÁREA DE PROGRAMAS	P	L	X	X	X	X			
1 1 1 1 0	CARGA DA ÁREA DE DADOS	D	L	X	X	X	X			
1 1 1 1 1	TROCA	D	L	X	X	X	X			

TABELA DE CONVERSÃO

4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4 096	1	256	1	16	1	1
2	8 192	2	512	2	32	2	2
3	12 288	3	768	3	48	3	3
4	16 384	4	1 024	4	64	4	4
5	20 480	5	1 280	5	80	5	5
6	24 576	6	1 536	6	96	6	6
7	28 672	7	1 792	7	112	7	7
8	32 768	8	2 048	8	128	8	8
9	36 864	9	2 304	9	144	9	9
A	40 960	A	2 560	A	160	A	10
B	45 056	B	2 816	B	176	B	11
C	49 152	C	3 072	C	192	C	12
D	53 248	D	3 328	D	208	D	13
E	57 344	E	3 584	E	224	E	14
F	61 440	F	3 840	F	240	F	15
0123		4567		0123		4567	
BYTE				BYTE			

REFERÊNCIAS

1. Ellis, Robert A., "Modular Computer Systems" COMPCON 72 DIGEST

2. Bell, C.Gordon and Allen Newell, "Computer Structures": Reading and Examples", McGraw-Hill, N.Y. - 1971

3. Dinman, B., "The Direct Function Processor Concept for System Control", Computer Design, March, 1970.

4. Altmam, Laurence; "Single-chip microprocessor open up a new world of application", Electronics April 18, 1974.

5. Samir S. Husson; "Microprogramming: Principles and Practices" Prentice-Hall, New-York, 1970.

6. Bell C. Gordon; J. Grasson and A. Newell, "Designimg Computers and Digital Systems", Digital Press, 1972.

7. Grasson, J; C.G Bell and J. Eggert. "The Commercialization of Register Transfer Modules", IEE Computer, October, 1973.

8. Clark, Wesley A. and Charles E. Molnar; "The promise of Macro modular Systems", COMPCON 72 Digest of Papers, IEE Computer society - 1972 pp. 309-312.

9. Robinson, D.M. "Digital System Design with control Modules", IEE computer society international conference, COMPCON 73, March, 1973 pp. 207-210.

10. Richards, Charles L., "An easy way to design complex program controllers", Electronics, february 1, - 1973.

11. Gschwind, Hans W.; "Design of Digital Computers" Springer-Verlag, New-York, - 1967.

12. C.V. Ramamorthy and M. Tsuchiya; "Analyses of microprogrammed processor trade-off", COMPCON 72.

13. Kenneth J. Thurber, E. Douglas Jensen, Larry A. Jack, Larry L. Kinney, Peter C. Patton, and Lynn C. Anderson:"A systematic approach to the design of digital bussing structures", Fall Joint Computer Conference - 1972.

14. Fairchild, Application note; "Philosophy and Design of MSI" 1971.

15. Fregni, Edson; "PRojeto Lógico da unidade de controle de um minicomputador"; Dissertação de mestrado - EPUSP. - 1972.

FD-87
e.2 PEL
AUTOR: Ruggiero, Wilson Vicente
TÍTULO: Projeto de um processador central
micropro...

FD-87
e.2 PEL
AUTOR: Ruggiero, Wilson Vicente
TITULO: Projeto de um processador central
microprogramado.

DATA DA RETIRADA	ASSINATURAS	DATA DA DEVOLUÇÃO

FD-87
E.2 PEL
Ruggiero, Wilson Vicente
Projeto de um processador central
microprogramado.